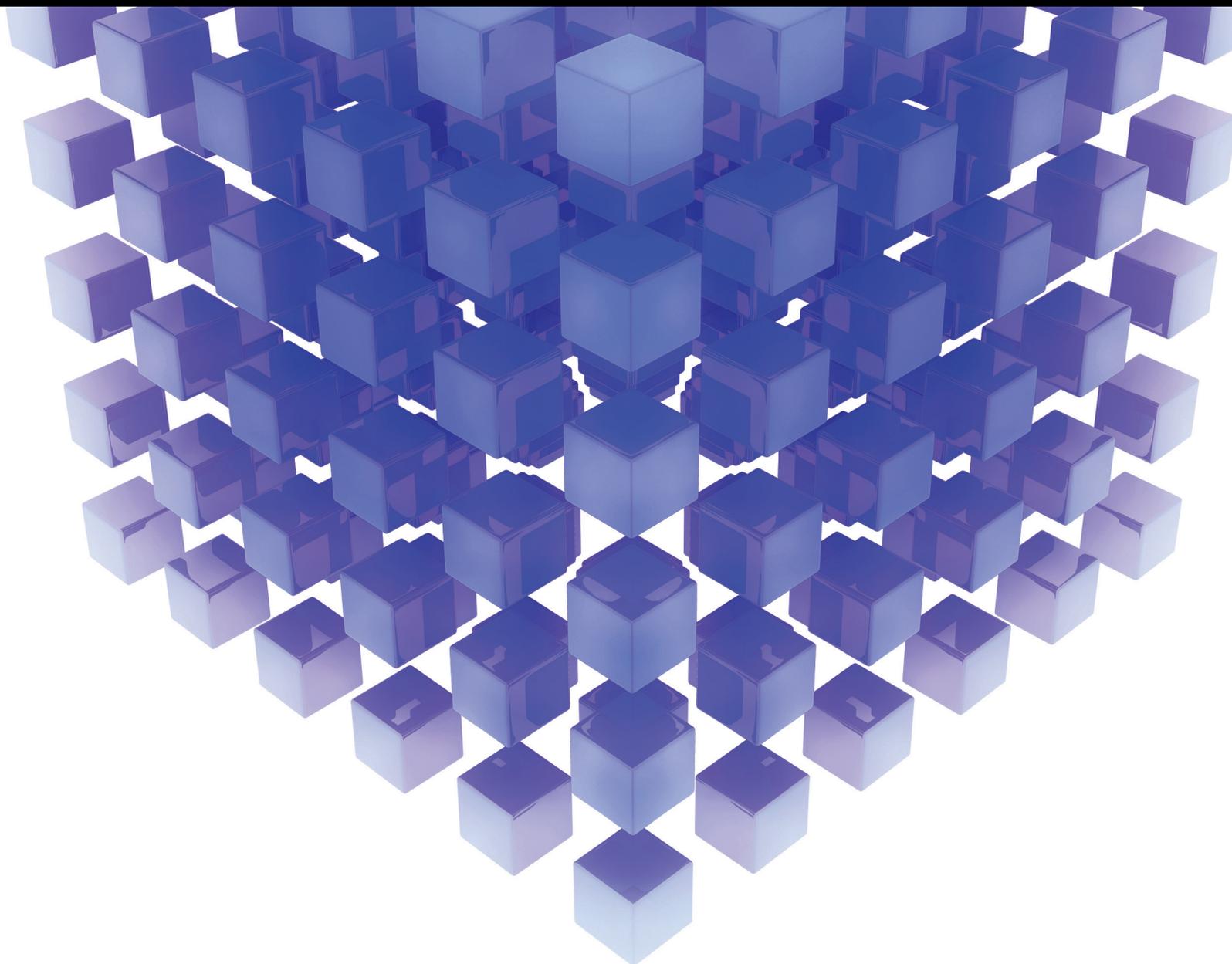


# Extreme Learning Machine on High Dimensional and Large Data Applications

Guest Editors: Zhiping Lin, Jiuwen Cao, Tao Chen, Yi Jin, Zhan-li Sun,  
and Amaury Lendasse





---

# **Extreme Learning Machine on High Dimensional and Large Data Applications**

Mathematical Problems in Engineering

---

## **Extreme Learning Machine on High Dimensional and Large Data Applications**

Guest Editors: Zhiping Lin, Jiuwen Cao, Tao Chen, Yi Jin, Zhan-li Sun, and Amaury Lendasse



---

Copyright © 2015 Hindawi Publishing Corporation. All rights reserved.

This is a special issue published in “Mathematical Problems in Engineering.” All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

## Editorial Board

- Mohamed Abd El Aziz, Egypt  
Farid Abed-Meraim, France  
Silvia Abrahão, Spain  
Paolo Addresso, Italy  
Claudia Adduce, Italy  
Ramesh Agarwal, USA  
Juan C. Agüero, Australia  
Ricardo Aguilar-López, Mexico  
Tarek Ahmed-Ali, France  
Hamid Akbarzadeh, Canada  
Muhammad N. Akram, Norway  
Mohammad-Reza Alam, USA  
Salvatore Alfonzetti, Italy  
Francisco Alhama, Spain  
Juan A. Almendral, Spain  
Saiied Aminossadati, Australia  
Lionel Amodeo, France  
Igor Andrianov, Germany  
Sebastian Anita, Romania  
Renata Archetti, Italy  
Felice Arena, Italy  
Sabri Arik, Turkey  
Fumihiko Ashida, Japan  
Hassan Askari, Canada  
Mohsen Asle Zaeem, USA  
Francesco Aymerich, Italy  
Seungik Baek, USA  
Khaled Bahlali, France  
Laurent Bako, France  
Stefan Balint, Romania  
Alfonso Banos, Spain  
Roberto Baratti, Italy  
Martino Bardi, Italy  
Azeddine Beghdadi, France  
Abdel-Hakim Bendada, Canada  
Ivano Benedetti, Italy  
Elena Benvenuti, Italy  
Jamal Berakdar, Germany  
Enrique Berjano, Spain  
Jean-Charles Beugnot, France  
Simone Bianco, Italy  
David Bigaud, France  
Jonathan N. Blakely, USA  
Paul Bogdan, USA  
Daniela Boso, Italy
- Abdel-Ouahab Boudraa, France  
Francesco Braghin, Italy  
Michael J. Brennan, UK  
Gunther Brenner, Germany  
Maurizio Brocchini, Italy  
Julien Bruchon, France  
Javier Buldú, Spain  
Tito Busani, USA  
Pierfrancesco Cacciola, UK  
Salvatore Caddemi, Italy  
Jose E. Capilla, Spain  
Ana Carpio, Spain  
Miguel E. Cerrolaza, Spain  
Mohammed Chadli, France  
Gregory Chagnon, France  
Ching-Ter Chang, Taiwan  
Michael J. Chappell, UK  
Kacem Chehdi, France  
Chunlin Chen, China  
Xinkai Chen, Japan  
Francisco Chicano, Spain  
Hung-Yuan Chung, Taiwan  
Joaquim Ciurana, Spain  
John D. Clayton, USA  
Carlo Cosentino, Italy  
Paolo Crippa, Italy  
Erik Cuevas, Mexico  
Peter Dabnichki, Australia  
Luca D'Acerno, Italy  
Weizhong Dai, USA  
Purushothaman Damodaran, USA  
Farhang Daneshmand, Canada  
Fabio De Angelis, Italy  
Stefano de Miranda, Italy  
Filippo de Monte, Italy  
Xavier Delorme, France  
Luca Deseri, USA  
Yannis Dimakopoulos, Greece  
Zhengtao Ding, UK  
Ralph B. Dinwiddie, USA  
Mohamed Djemai, France  
Alexandre B. Dolgui, France  
George S. Dulikravich, USA  
Bogdan Dumitrescu, Finland  
Horst Ecker, Austria
- Karen Egiazarian, Finland  
Ahmed El Hajjaji, France  
Fouad Erchiqui, Canada  
Anders Eriksson, Sweden  
Giovanni Falsone, Italy  
Hua Fan, China  
Yann Favennec, France  
Giuseppe Fedele, Italy  
Roberto Fedele, Italy  
Jacques Ferland, Canada  
Jose R. Fernandez, Spain  
Simme Douwe Flapper, The Netherlands  
Thierry Floquet, France  
Eric Florentin, France  
Francesco Franco, Italy  
Tomonari Furukawa, USA  
Mohamed Gadala, Canada  
Matteo Gaeta, Italy  
Zoran Gajic, USA  
Ciprian G. Gal, USA  
Ugo Galvanetto, Italy  
Akemi Gálvez, Spain  
Rita Gamberini, Italy  
Maria Gandarias, Spain  
Arman Ganji, Canada  
Xin-Lin Gao, USA  
Zhong-Ke Gao, China  
Giovanni Garcea, Italy  
Fernando Garcia, Spain  
Laura Gardini, Italy  
Alessandro Gasparetto, Italy  
Vincenzo Gattulli, Italy  
Jürgen Geiser, Germany  
Oleg V. Gendelman, Israel  
Mergen H. Ghayesh, Australia  
Anna M. Gil-Lafuente, Spain  
Hector Gómez, Spain  
Rama S. R. Gorla, USA  
Oded Gottlieb, Israel  
Antoine Grall, France  
Jason Gu, Canada  
Quang Phuc Ha, Australia  
Ofer Hadar, Israel  
Masoud Hajarian, Iran  
Frédéric Hamelin, France

Zhen-Lai Han, China  
Thomas Hanne, Switzerland  
Xiao-Qiao He, China  
María I. Herreros, Spain  
Vincent Hilaire, France  
Eckhard Hitzer, Japan  
Jaromir Horacek, Czech Republic  
Muneo Hori, Japan  
András Horváth, Italy  
Gordon Huang, Canada  
Sajid Hussain, Canada  
Asier Ibeas, Spain  
Giacomo Innocenti, Italy  
Emilio Insfran, Spain  
Nazrul Islam, USA  
Payman Jalali, Finland  
Reza Jazar, Australia  
Khalide Jbilou, France  
Linni Jian, China  
Bin Jiang, China  
Zhongping Jiang, USA  
Ningde Jin, China  
Grand R. Joldes, Australia  
Joaquim Joao Judice, Portugal  
Tadeusz Kaczorek, Poland  
Tamas Kalmar-Nagy, Hungary  
Tomasz Kapitaniak, Poland  
Haranath Kar, India  
Konstantinos Karamanos, Belgium  
C. M. Khaliq, South Africa  
Do Wan Kim, Korea  
Nam-Il Kim, Korea  
Oleg Kirillov, Germany  
Manfred Krafczyk, Germany  
Frederic Kratz, France  
Jurgen Kurths, Germany  
Kyandoghere Kyamakya, Austria  
Davide La Torre, Italy  
Risto Lahdelma, Finland  
Hak-Keung Lam, UK  
Antonino Laudani, Italy  
Aime' Lay-Ekuakille, Italy  
Marek Lefik, Poland  
Yaguo Lei, China  
Thibault Lemaire, France  
Stefano Lenci, Italy  
Roman Lewandowski, Poland  
Qing Q. Liang, Australia  
Panos Liatsis, UK  
Wanquan Liu, Australia  
Yan-Jun Liu, China  
Peide Liu, China  
Peter Liu, Taiwan  
Jean J. Loiseau, France  
Paolo Lonetti, Italy  
Luis M. López-Ochoa, Spain  
Vassilios C. Loukopoulos, Greece  
Valentin Lychagin, Norway  
F. M. Mahomed, South Africa  
Yassir T. Makkawi, UK  
Noureddine Manamanni, France  
Didier Maquin, France  
Paolo Maria Mariano, Italy  
Benoit Marx, France  
Gérard A. Maugin, France  
Driss Mehdi, France  
Roderick Melnik, Canada  
Pasquale Memmolo, Italy  
Xiangyu Meng, Canada  
Jose Merodio, Spain  
Luciano Mescia, Italy  
Laurent Mevel, France  
Yuri V. Mikhlin, Ukraine  
Aki Mikkola, Finland  
Hiroyuki Mino, Japan  
Pablo Mira, Spain  
Vito Mocella, Italy  
Roberto Montanini, Italy  
Gisele Mophou, France  
Rafael Morales, Spain  
Aziz Moukrim, France  
Emiliano Mucchi, Italy  
Domenico Mundo, Italy  
Jose J. Muñoz, Spain  
Giuseppe Muscolino, Italy  
Marco Mussetta, Italy  
Hakim Naceur, France  
Hassane Naji, France  
Dong Ngoduy, UK  
Tatsushi Nishi, Japan  
Ben T. Nohara, Japan  
Mohammed Nouari, France  
Mustapha Nourelfath, Canada  
Sotiris K. Ntouyas, Greece  
Roger Ohayon, France  
Mitsuhiro Okayasu, Japan  
Eva Onaindia, Spain  
Javier Ortega-Garcia, Spain  
Alejandro Ortega-Moñux, Spain  
Naohisa Otsuka, Japan  
Erika Ottaviano, Italy  
Alkiviadis Paipetis, Greece  
Alessandro Palmeri, UK  
Anna Pandolfi, Italy  
Elena Panteley, France  
Manuel Pastor, Spain  
Pubudu N. Pathirana, Australia  
Francesco Pellicano, Italy  
Mingshu Peng, China  
Haipeng Peng, China  
Zhike Peng, China  
Marzio Pennisi, Italy  
Matjaz Perc, Slovenia  
Francesco Pesavento, Italy  
Maria do Rosário Pinho, Portugal  
Antonina Pirrotta, Italy  
Vicent Pla, Spain  
Javier Plaza, Spain  
Jean-Christophe Ponsart, France  
Mauro Pontani, Italy  
Stanislav Potapenko, Canada  
Sergio Preidikman, USA  
Christopher Pretty, New Zealand  
Carsten Proppe, Germany  
Luca Pugi, Italy  
Yuming Qin, China  
Dane Quinn, USA  
Jose Ragot, France  
K. Ramamani Rajagopal, USA  
Gianluca Ranzi, Australia  
Sivaguru Ravindran, USA  
Alessandro Reali, Italy  
Giuseppe Rega, Italy  
Oscar Reinoso, Spain  
Nidhal Rezg, France  
Ricardo Riaza, Spain  
Gerasimos Rigatos, Greece  
José Rodellar, Spain  
Rosana Rodriguez-Lopez, Spain  
Ignacio Rojas, Spain  
Carla Roque, Portugal  
Aline Roumy, France  
Debasish Roy, India  
Rubén Ruiz García, Spain

Antonio Ruiz-Cortes, Spain  
Ivan D. Rukhlenko, Australia  
Mazen Saad, France  
Kishin Sadarangani, Spain  
Mehrdad Saif, Canada  
Miguel A. Salido, Spain  
Roque J. Saltarén, Spain  
Francisco J. Salvador, Spain  
Alessandro Salvini, Italy  
Maura Sandri, Italy  
Miguel A. F. Sanjuan, Spain  
Juan F. San-Juan, Spain  
Roberta Santoro, Italy  
Ilmar Ferreira Santos, Denmark  
José A. Sanz-Herrera, Spain  
Nickolas S. Sapidis, Greece  
Evangelos J. Sapountzakis, Greece  
Themistoklis P. Sapsis, USA  
Andrey V. Savkin, Australia  
Valery Sbitnev, Russia  
Thomas Schuster, Germany  
Mohammed Seaid, UK  
Lotfi Senhadji, France  
Joan Serra-Sagrasta, Spain  
Leonid Shaikhet, Ukraine  
Hassan M. Shanechi, USA  
Sanjay K. Sharma, India  
Bo Shen, Germany  
Babak Shotorban, USA  
Zhan Shu, UK  
Dan Simon, USA  
Luciano Simoni, Italy  
Christos H. Skiadas, Greece  
Michael Small, Australia

Francesco Soldovieri, Italy  
Raffaele Solimene, Italy  
Ruben Specogna, Italy  
Sri Sridharan, USA  
Ivanka Stamova, USA  
Yakov Strelniker, Israel  
Sergey A. Suslov, Australia  
Thomas Svensson, Sweden  
Andrzej Swierniak, Poland  
Yang Tang, Germany  
Sergio Teggi, Italy  
Roger Temam, USA  
Alexander Timokha, Norway  
Rafael Toledo, Spain  
Gisella Tomasini, Italy  
Francesco Tornabene, Italy  
Antonio Tornambe, Italy  
Fernando Torres, Spain  
Fabio Tramontana, Italy  
Sébastien Tremblay, Canada  
Irina N. Trendafilova, UK  
George Tsiatas, Greece  
Antonios Tsourdos, UK  
Vladimir Turetsky, Israel  
Mustafa Tutar, Spain  
Efstratios Tzirtzilakis, Greece  
Francesco Ubertini, Italy  
Filippo Ubertini, Italy  
Hassan Ugail, UK  
Giuseppe Vairo, Italy  
Kuppalapalle Vajravelu, USA  
Robertt A. Valente, Portugal  
Raoul van Loon, UK  
Pandian Vasant, Malaysia

Miguel E. Vázquez-Méndez, Spain  
Josep Vehi, Spain  
Kalyana C. Veluvolu, Korea  
Fons J. Verbeek, The Netherlands  
Franck J. Vernerey, USA  
Georgios Veronis, USA  
Anna Vila, Spain  
Rafael J. Villanueva, Spain  
U. E. Vincent, UK  
Mirko Viroli, Italy  
Michael Vynnycky, Sweden  
Yan-Wu Wang, China  
Junwu Wang, China  
Shuming Wang, Singapore  
Yongqi Wang, Germany  
Jeroen A. S. Witteveen, The Netherlands  
Yuqiang Wu, China  
Dash Desheng Wu, Canada  
Guangming Xie, China  
Xuejun Xie, China  
Gen Qi Xu, China  
Hang Xu, China  
Xinggong Yan, UK  
Luis J. Yebra, Spain  
Peng-Yeng Yin, Taiwan  
Ibrahim Zeid, USA  
Huaguang Zhang, China  
Qingling Zhang, China  
Jian Guo Zhou, UK  
Quanxin Zhu, China  
Mustapha Zidi, France  
Alessandro Zona, Italy

# Contents

**Extreme Learning Machine on High Dimensional and Large Data Applications**, Zhiping Lin, Jiuwen Cao, Tao Chen, Yi Jin, Zhan-li Sun, and Amaury Lendasse  
Volume 2015, Article ID 624903, 2 pages

**Fault Tolerance Automotive Air-Ratio Control Using Extreme Learning Machine Model Predictive Controller**, Pak Kin Wong, Hang Cheong Wong, Chi Man Vong, Tong Meng Iong, Ka In Wong, and Xianghui Gao  
Volume 2015, Article ID 317142, 10 pages

**Modeling, Prediction, and Control of Heating Temperature for Tube Billet**, Yachun Mao, Dong Xiao, and Dapeng Niu  
Volume 2015, Article ID 576813, 10 pages

**Efficient ELM-Based Two Stages Query Processing Optimization for Big Data**, Linlin Ding, Yu Liu, Baoyan Song, and Junchang Xin  
Volume 2015, Article ID 236084, 12 pages

**Distributed Learning over Massive XML Documents in ELM Feature Space**, Xin Bi, Xiangguo Zhao, Guoren Wang, Zhen Zhang, and Shuang Chen  
Volume 2015, Article ID 923097, 13 pages

**Multiclass AdaBoost ELM and Its Application in LBP Based Face Recognition**, Yunliang Jiang, Yefeng Shen, Yong Liu, and Weicong Liu  
Volume 2015, Article ID 918105, 9 pages

**Daily Human Physical Activity Recognition Based on Kernel Discriminant Analysis and Extreme Learning Machine**, Wendong Xiao and Yingjie Lu  
Volume 2015, Article ID 790412, 8 pages

**Real-Time and Accurate Indoor Localization with Fusion Model of Wi-Fi Fingerprint and Motion Particle Filter**, Xinlong Jiang, Yiqiang Chen, Junfa Liu, Dingjun Liu, Yang Gu, and Zhenyu Chen  
Volume 2015, Article ID 545792, 13 pages

**Research on Three-dimensional Motion History Image Model and Extreme Learning Machine for Human Body Movement Trajectory Recognition**, Zheng Chang, Xiaojuan Ban, Qing Shen, and Jing Guo  
Volume 2015, Article ID 528190, 15 pages

**An ELM-Based Approach for Estimating Train Dwell Time in Urban Rail Traffic**, Wen-jun Chu, Xing-chen Zhang, Jun-hua Chen, and Bin Xu  
Volume 2015, Article ID 473432, 9 pages

**Distance Based Multiple Kernel ELM: A Fast Multiple Kernel Learning Approach**, Chengzhang Zhu, Xinwang Liu, Qiang Liu, Yuewei Ming, and Jianping Yin  
Volume 2015, Article ID 372748, 9 pages

**Deep Extreme Learning Machine and Its Application in EEG Classification**, Shifei Ding, Nan Zhang, Xinzheng Xu, Lili Guo, and Jian Zhang  
Volume 2015, Article ID 129021, 11 pages

**The Optimisation for Local Coupled Extreme Learning Machine Using Differential Evolution,**

Yanpeng Qu and Ansheng Deng

Volume 2015, Article ID 946292, 9 pages

**Detecting Copy Directions among Programs Using Extreme Learning Machines,** Bin Wang,

Xiaochun Yang, and Guoren Wang

Volume 2015, Article ID 793697, 15 pages

**Online Sequential Prediction for Nonstationary Time Series with New Weight-Setting Strategy Using Extreme Learning Machine,** Wentao Mao, Jinwan Wang, Liyun Wang, and Mei Tian

Volume 2015, Article ID 484093, 13 pages

**Data-Driven Dynamic Modeling for Prediction of Molten Iron Silicon Content Using ELM with Self-Feedback,** Ping Zhou, Meng Yuan, Hong Wang, and Tianyou Chai

Volume 2015, Article ID 326160, 11 pages

**An ELM Based Online Soft Sensing Approach for Alumina Concentration Detection,** Sen Zhang,

Xi Chen, and Yixin Yin

Volume 2015, Article ID 268132, 8 pages

**Two-Dimensional Extreme Learning Machine,** Bo Jia, Dong Li, Zhisong Pan, and Guyu Hu

Volume 2015, Article ID 491587, 8 pages

**Extreme Learning Machine Assisted Adaptive Control of a Quadrotor Helicopter,** Yu Zhang,

Zheng Fang, and Hongbo Li

Volume 2015, Article ID 905184, 12 pages

**A Robust AdaBoost.RT Based Ensemble Extreme Learning Machine,** Pengbo Zhang and Zhixin Yang

Volume 2015, Article ID 260970, 12 pages

**Weibo Information Propagation Dissemination Based on User Behavior Using ELM,**

Huilin Liu and Yao Li

Volume 2015, Article ID 876218, 11 pages

**Particle Swarm Optimization Based Selective Ensemble of Online Sequential Extreme Learning Machine,** Yang Liu, Bo He, Diya Dong, Yue Shen, Tianhong Yan, Rui Nian, and Amaury Lendasse

Volume 2015, Article ID 504120, 10 pages

**One-Class Classification with Extreme Learning Machine,** Qian Leng, Honggang Qi, Jun Miao, Wentao Zhu, and Guiping Su

Volume 2015, Article ID 412957, 11 pages

**Anomaly Detection via Midlevel Visual Attributes,** Tan Xiao, Chao Zhang, and Hongbin Zha

Volume 2015, Article ID 343869, 15 pages

**Improving ELM-Based Service Quality Prediction by Concise Feature Extraction,** Yuhai Zhao, Ying Yin,

Gang Sheng, Bin Zhang, and Guoren Wang

Volume 2015, Article ID 325192, 15 pages

# Contents

---

**Extreme Learning Machine for Reservoir Parameter Estimation in Heterogeneous Sandstone Reservoir,**  
Jianhua Cao, Jucheng Yang, Yan Wang, Dan Wang, and Yancui Shi  
Volume 2015, Article ID 287816, 10 pages

**Keyword Search over Probabilistic XML Documents Based on Node Classification,** Yue Zhao, Ye Yuan,  
and Guoren Wang  
Volume 2015, Article ID 210961, 11 pages

**Quasilinear Extreme Learning Machine Model Based Internal Model Control for Nonlinear Process,**  
Dazi Li, Qianwen Xie, and Qibing Jin  
Volume 2015, Article ID 181389, 9 pages

**Sample-Based Extreme Learning Machine with Missing Data,** Hang Gao, Xin-Wang Liu, Yu-Xing Peng,  
and Song-Lei Jian  
Volume 2015, Article ID 145156, 11 pages

**Optimization ELM Based on Rough Set for Predicting the Label of Military Simulation Data,**  
Xiao-jian Ding and Ming Lei  
Volume 2014, Article ID 706178, 8 pages

**Modeling and Optimization for Piercing Efficiency and Energy Consumption Based on Mean Value  
Substaged KELM-PLS and GA Method,** Dong Xiao and Jichun Wang  
Volume 2014, Article ID 132654, 12 pages

## Editorial

# Extreme Learning Machine on High Dimensional and Large Data Applications

Zhiping Lin,<sup>1</sup> Jiuwen Cao,<sup>2</sup> Tao Chen,<sup>3</sup> Yi Jin,<sup>4</sup> Zhan-Li Sun,<sup>5</sup> and Amaury Lendasse<sup>6,7</sup>

<sup>1</sup>*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798*

<sup>2</sup>*Key Lab for IOT and Information Fusion Technology of Zhejiang, Hangzhou Dianzi University, Zhejiang 310018, China*

<sup>3</sup>*Department of Visual Computing, Institute for Infocomm Research, Agency for Science, Technology and Research, 1 Fusionopolis Way, No. 21-01 Connexis (South Tower), Singapore 138632*

<sup>4</sup>*School of Computer and Information Technology, Beijing Jiaotong University, No. 3 Shangyuan, Haidian District, Beijing 100044, China*

<sup>5</sup>*School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China*

<sup>6</sup>*Department of Mechanical and Industrial Engineering and The Iowa Informatics Initiative, The University of Iowa, Iowa City, IA 52242-1527, USA*

<sup>7</sup>*Arcada University of Applied Sciences, 00560 Helsinki, Finland*

Correspondence should be addressed to Zhiping Lin; [ezplin@ntu.edu.sg](mailto:ezplin@ntu.edu.sg)

Received 29 March 2015; Accepted 29 March 2015

Copyright © 2015 Zhiping Lin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Extreme learning machine (ELM) has been developed for single hidden layer feedforward neural networks (SLFNs) training in the past decade. ELM theory claims that the hidden node parameters of SLFNs can be randomly generated and need not be updated. All the hidden node parameters are independent of the target functions or the training datasets. Benefitting from the tuning-free framework, ELM not only learns up to thousand times faster than conventional gradient descent methods for SLFNs and the support vector machine (SVM) but also preserves a reasonable generalization performance. For most applications, it has been shown that the learning phase of ELM is almost real-time processing in an ordinary PC. Therefore, ELM shows superiority over conventional gradient based methods and SVM on high dimensional applications and large data processing. This is especially important since nowadays data are explosive with the rapid development of the Internet, computer, and electronic equipment.

In the light of these considerations, this special issue was launched. We received papers from worldwide researchers in interdisciplinary research fields and high quality peer reviewed papers are selected to include in this special issue.

An overview paper which summarizes the recent developments on ELM and its applications in high dimensional and large data processing presented by the Guest Editors J. Cao and Z. Lin is also published in the special issue. These papers present the recent developments in ELM algorithms applications in high dimensional and large data. The accepted papers can be broadly categorized into six groups: image and video processing, target tracking and predictions, control and estimation application, ELM algorithms, web and document applications, and other applications. In the following, we give a brief description of the papers in each of the six groups.

In image and video processing, W. Xiao and Y. Lu developed a highly efficient approach for human activity recognition based on ELM and using only one triaxial accelerometer. S. Ding et al. combined multilayer ELM (MLELM) and ELM with kernel (KELM) to put forward deep extreme learning machine (DELME) and apply it to EEG classification. T. Xiao et al. proposed a novel automatic anomaly detection approach with ELM-based visual attribute and Spatiotemporal Pyramid (STP). Y. Jiang et al. developed a new boosting ELM named MAELM, which applies the Multiclass AdaBoost in ELM ensemble to directly solve

multiclass classification problem. Z. Chang et al. utilized the three-dimensional motion history image and the ELM to realize the recognition of body movement trajectory robustly, efficiently, and accurately.

In target tracking and predictions, X. Jiang et al. proposed fusion location framework with particle filter using Wi-Fi signals and motion sensors. The ELM regression algorithm is used to predict position based on motion sensors, and Wi-Fi fingerprint location is used to solve the error accumulation of motion sensors. J. Cao et al. adopted the ELM-based predictors for the prediction of porosity and permeability in heterogeneous sandstone reservoir of Permian formation in Yanqi survey. P. Zhou et al. developed a data-driven dynamic modeling method for online prediction of silicon content using improved ELM with the help of principle component analysis (PCA). W. Mao et al. proposed an online sequential ELM with the new weighted strategy for nonstationary time series prediction. This strategy ranks the samples' importance by means of the Leave-One-Out (LOO) error of each new added sample and then assigns various weights. Y. Zhao et al. investigated an active service quality prediction method based on ELM by concise feature extraction. An efficient prefix tree based mining algorithm together with some effective pruning rules is developed to mine the EC rule.

In control and estimation applications, D. Xiao and J. Wang developed a mean value staged kernel ELM combining with the partial least square method (KELM-PLS) to optimize the piercing efficiency and energy consumption prediction model. Y. Mao et al. proposed an online sequential ELM dynamic recursive partial least square approach (OS-ELM-DRPLS) to establish a prediction model for the final temperature of a tube billet. W. Chu et al. applied the original ELM to estimate train dwell time in urban rail traffic. P. K. Wong et al. employed the kernel ELM algorithm as a model predictive controller to build a backup air-ratio model. D. Li et al. introduced a new quasilinear model based ELM (QL-ELM) algorithm for internal model control with nonlinearity. Y. Zhang et al. proposed a novel intelligent control design based on ELM to assist the control of a quadrotor helicopter.

In developments of ELM algorithm, P. Zhang and Z. Yang developed a new hybrid machine learning method called robust AdaBoost.RT based ensemble ELM (RAE-ELM) for real-world regression problems. C. Zhu et al. proposed a distance based multiple kernel ELM (DBMK-ELM), which provides a two-stage multiple kernel learning approach with high efficiency. B. Jia et al. presented the two-dimensional ELM (2DELME) by dealing with matrix data directly. Unlike original ELM that handles vectors, 2DELME takes the matrices as input features without vectorization. Q. Leng et al. proposed a simple and efficient one-class classifier based on ELM to tackle the slow learning speed in autoencoder neural network. Y. Qu and A. Deng implemented the evolutionary local coupled ELM (ELC-ELM) to further refine the weight searching space of LC-ELM. Y. Liu et al. developed a novel particle swarm optimization based selective ensemble (PSOSEN) of online sequential ELM.

In web and document applications, H. Liu and Y. Li applied a new feature to analyze user behavior and information dissemination and used ELM to predict behaviors of

users. Y. Zhao et al. proposed a keyword search measure on probabilistic extensible markup language (XML) data based on ELM. B. Wang et al. constructed feature space to describe features of every two programs with possible plagiarism relationship by employing ELM. X. Bi et al. presented a solution to the distributed learning over massive XML documents. The new method provides distributed conversion of XML documents into the representation model in parallel based on MapReduce. A distributed learning component based on ELM is then used for mining tasks of classification or clustering.

In other applications with ELM, H. Gao et al. investigated two sample-based ELM algorithms in handling the problem of missing data in regressions and classifications. L. Ding et al. developed an efficient ELM-based two stages query processing optimization model for big data. S. Zhang et al. utilized the modified ELM approach to develop a soft-sensing method for online detection of the alumina concentration. X. Ding and M. Lei combined the constrained-optimization based ELM with the rough set theory for military data classification and labeling.

## Acknowledgments

As the Guest Editors of the special issue, we would like to express our gratitude to all the authors who have contributed their work to our special issue. We also would like to express our sincere appreciation to the reviewers for their valuable and insightful comments on the submitted manuscripts to our special issue.

*Zhiping Lin  
Jiuwen Cao  
Tao Chen  
Yi Jin  
Zhan-Li Sun  
Amaury Lendasse*

## Research Article

# Fault Tolerance Automotive Air-Ratio Control Using Extreme Learning Machine Model Predictive Controller

**Pak Kin Wong,<sup>1</sup> Hang Cheong Wong,<sup>1</sup> Chi Man Vong,<sup>2</sup> Tong Meng Iong,<sup>1</sup> Ka In Wong,<sup>1</sup> and Xianghui Gao<sup>1</sup>**

<sup>1</sup>Department of Electromechanical Engineering, University of Macau, Macau

<sup>2</sup>Department of Computer and Information Science, University of Macau, Macau

Correspondence should be addressed to Pak Kin Wong; [fstpkw@umac.mo](mailto:fstpkw@umac.mo)

Received 7 August 2014; Revised 27 September 2014; Accepted 28 September 2014

Academic Editor: Jiuwen Cao

Copyright © 2015 Pak Kin Wong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Effective air-ratio control is desirable to maintain the best engine performance. However, traditional air-ratio control assumes the lambda sensor located at the tail pipe works properly and relies strongly on the air-ratio feedback signal measured by the lambda sensor. When the sensor is warming up during cold start or under failure, the traditional air-ratio control no longer works. To address this issue, this paper utilizes an advanced modelling technique, kernel extreme learning machine (ELM), to build a backup air-ratio model. With the prediction from the model, a limited air-ratio control performance can be maintained even when the lambda sensor does not work. Such strategy is realized as fault tolerance control. In order to verify the effectiveness of the proposed fault tolerance air-ratio control strategy, a model predictive control scheme is constructed based on the kernel ELM backup air-ratio model and implemented on a real engine. Experimental results show that the proposed controller can regulate the air-ratio to specific target values within a satisfactory tolerance under external disturbance and the absence of air-ratio feedback signal from the lambda sensor. This implies that the proposed fault tolerance air-ratio control is a promising scheme to maintain air-ratio control performance when the lambda sensor is under failure or warming up.

## 1. Introduction

Vehicular emissions are the major source of gaseous pollutants that contribute to the harmful and negative effects on environment and human health. It has been reported in [1–4] that the increasing amount of vehicular emissions has led to hundred thousands of mortalities and billions of economic loss every year. To reduce the amount of toxic elements in vehicular emissions, three-way catalytic converter is currently the most effective after-treatment device. This device reduces the unburned hydrocarbons and carbon monoxide by oxidization and nitrogen oxides by reduction [5]. The conversion efficiency of the catalytic converter, however, depends highly on the air-ratio (also known as lambda). When the air-ratio is at the stoichiometric value (i.e., air-ratio = 1.0), the conversion efficiency can reach as high as 98%, but derivation of only 1% from stoichiometry can already result in 50% degradation on the converter. Therefore, for environmental purpose, the air-ratio is usually regulated to 1. Meanwhile,

as an important engine parameter, the air-ratio should also be controlled to different values for other situations [6]. For instance, if emissions are not concerned, the air-ratio can be regulated to around 0.95 to achieve the best engine power performance, whilst it is 1.05 to achieve the best brake-specific fuel consumption. Consequently, an effective air-ratio control system is necessary for engine system to maintain its best performance under various operating conditions.

Over the past decades, car manufacturers and researchers have developed many air-ratio control strategies [7–12]. Examples include the sliding mode control [7, 8], proportional-integral-derivative (PID) control [9], and neural-network-based model predictive control (MPC) [10–12]. Sliding mode control requires a very accurate mathematical definition of the engine model, but in general it is impossible to derive an exact engine dynamics model due to its highly nonlinear nature [13]. In most sliding mode air-ratio control studies, many assumptions have been made in the model derivation, and many coefficients are difficult to determine

for a real engine, so this strategy may not be suitable for practical use. Although PID control is the most widely used approach in practice, the calibration and tuning of the control parameters are very time-consuming and engine dependent. The tuned PID controller cannot deal with steady disturbance or any change in the engine conditions either. Thus, among these researches, the most appropriate and promising technique for air-ratio control is the neural-network-based MPC, due to its robustness to multivariable, time-varying, and delay systems like modern engine systems [12]. It is well-known that a reliable prediction model is the core component of the MPC, but the engine models developed in [10, 11] were only surrogate models. That is, the models were trained from the data generated by empirical equations rather than a real engine. Therefore, similar to the deficiency of sliding mode control approach, the neural-network prediction models in [10, 11] derived from data generated by empirical equations cannot effectively reflect the actual performance of real engines. In the most recent study of MPC air-control strategy [12], the prediction model for the controller was constructed based on experimental data rather than numerical data. Experimental results in [12] showed that the controller performance is superior to those of [10, 11] in real application. Nevertheless, one major concern for the control strategy in [12] is that the prediction model must rely on the real-time air-ratio signal measured from the lambda sensor located at the exhaust pipe of the engine. When the sensor is under failure or warming up during cold start, the controller becomes ineffective, resulting in poor control performance.

In fact, for most of the current available air-ratio control approaches, the air-ratio measurement must be acquired as the feedback to the controllers. Hence, the problem of lambda sensor failure must be addressed. Although on-board diagnostics for the lambda sensor has been a requirement for more than two decades [14] and any fault of the lambda sensor must be reflected through the “check engine” light on the instrument panel, the driver may not be aware of such fault and may not be willing to replace the lambda sensor when the car can still be driven without significant defect. In that case, the emissions and fuel consumption of that car are already significantly deteriorated. Therefore, maintaining a satisfactory air-ratio control performance when the lambda sensor is under failure or warming up during cold start is of great significance. This paper proposes to build a supplementary air-ratio model to compensate the lambda sensor, in which the measured air-ratio signal is not required as the model input.

From the open literature [15, 16], it is possible to predict the air-ratio without using the previous air-ratio signal. For instance, Gassenfeit and Powell [15] compared two algorithms that can predict the air-ratio from either the cylinder pressure time history patterns or the ratio of the cylinder pressure before and after combustion. Another example is the method described by Asik et al. [16], in which the air-ratio can be roughly estimated from induced crankshaft speed fluctuations. However, the quantities used in these algorithms, say, the in-cylinder pressure and delicate crankshaft speed fluctuation, are usually unavailable in normal vehicle engines because expensive sensors are required. Moreover,

as mentioned, empirical equations may not be suitable for real applications. Thus, by following the framework in [12], this study attempts to construct the air-ratio model from experimental data. Extreme learning machine (ELM) [17] is currently a popular and effective algorithm for training model from sample data. Many recent studies [18, 19] already showed that ELM is superior to other famous methods, such as neural-networks, least squares support vector machines, and relevance vector machine, in terms of generalization performance and computational load. ELM has been employed for various practical applications too [20, 21]. Thus, ELM is selected in this study to develop the supplementary air-ratio model.

Among so many variants of ELM, kernel-based ELM is adopted in this study to build the model. It is because, in kernel ELM, the random feature mapping is replaced with a kernel function, so randomness does not occur and the chance of result variations could be reduced [22]. In fact, the model built in [12] was based on an online variant of ELM, whose backbone is simply a basic ELM (i.e., the model is still an ELM with random feature mapping but can be updated when online data is provided). The reason why offline version of ELM is used in this paper instead of the online one is that the real-time data of air-ratio will not be available when the lambda sensor is malfunctioning. As both the air-ratio model from [12] and the proposed air-ratio model are made from the same basis (kernel ELM), a fair comparison can be made to evaluate their performances.

In order to verify the effectiveness of the ELM supplementary air-ratio model, this paper also proposes a non-linear MPC algorithm for air-ratio control, which utilizes a switch to toggle between the lambda sensor signals and the ELM supplementary air-ratio model predictions. When the lambda sensor works well, the air-ratio measurement will be used; when the lambda sensor is under failure, the backup air-ratio model will be used. The MPC under such strategy is called fault tolerance controller (FTC), which is a novel nontrivial application of ELM. Based on the multiple-step-ahead air-ratio predictions, a control signal is obtained to regulate the air-ratio to trace the desired values. The proposed FTC is also compared with the typical air-ratio control techniques, including online sequential extreme learning machine model predictive controller (OEMPC) [12], diagonal recurrent neural-network model predictive controller (DNMPC) [10], and traditional open-loop air-ratio control system, to evaluate its performance. The concept of kernel-based ELM is provided in Section 2. The construction and evaluation of the ELM supplementary air-ratio model are given in Section 3. The detail of the FTC design is presented in Section 4. Experimental implementation and evaluation of the proposed FTC and OEMPC are provided in Section 5.

## 2. Kernel-Based Extreme Learning Machine

Kernel-based ELM is a learning scheme for single-hidden-layer feedforward network, with the use of kernel [17]. Considering a set of  $N$  training samples  $\mathcal{D} = \{(\mathbf{x}_i, t_i), i = 1, \dots, N\}$ , with each  $\mathbf{x}_i$  being a  $d$  dimensional input vector

and  $t_i$  as the target scalar output, a single-hidden-layer feedforward network with  $L$  hidden nodes can be written as

$$\sum_{k=1}^L \beta_k h_k(\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta} = t_i, \quad i = 1, \dots, N, \quad (1)$$

where  $\mathbf{h}(\mathbf{x}_i) = [h_1(\mathbf{x}_i), \dots, h_L(\mathbf{x}_i)]$  is the feature mapping output with respect to  $\mathbf{x}_i$  and  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_L]^T$  is the output weight vector. In kernel-based ELM, this weight vector is determined by minimizing both the norm of the weight vector and the training error. The corresponding optimization problem is

$$\begin{aligned} \text{Minimize: } & \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\theta_i\|^2 \\ \text{Subject to: } & \theta_i = t_i - \mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta}, \quad i = 1, \dots, N, \end{aligned} \quad (2)$$

where  $C$  is a user-specified penalty term for regularization purpose.

Then, based on the Karush-Kuhn-Tucker theorem, optimizing (2) is equal to solving the following dual optimization problem:

$$\begin{aligned} \mathcal{L}_{\text{ELM}} = & \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\theta_i\|^2 \\ & - \sum_{i=1}^N \alpha_i (\mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta} - t_i + \theta_i), \end{aligned} \quad (3)$$

where  $\alpha_i$  is the Lagrange multiplier.

By taking derivatives on (3), the following conditions are obtained:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{ELM}}}{\partial \boldsymbol{\beta}} = 0 & \longrightarrow \boldsymbol{\beta} = \sum_{i=1}^N \alpha_i \mathbf{h}(\mathbf{x}_i)^T \longrightarrow \boldsymbol{\beta} = \mathbf{H}^T \boldsymbol{\alpha}, \\ \frac{\partial \mathcal{L}_{\text{ELM}}}{\partial \theta_i} = 0 & \longrightarrow \alpha_i = C \theta_i, \quad i = 1, \dots, N, \\ \frac{\partial \mathcal{L}_{\text{ELM}}}{\partial \alpha_i} = 0 & \longrightarrow \mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta} - t_i + \theta_i = 0, \quad i = 1, \dots, N, \end{aligned} \quad (4)$$

where

$$\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^T, \quad \mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \cdots & h_L(\mathbf{x}_N) \end{bmatrix}_{N \times L}. \quad (5)$$

By combining the conditions in (4) and eliminating the Lagrange multipliers  $\alpha_i$ , the optimal weight vector could be calculated as

$$\boldsymbol{\beta} = \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}, \quad (6)$$

where  $\mathbf{T} = [t_1, \dots, t_N]^T$  and  $\mathbf{I}$  is the identity matrix.

With (6), the output function of the network for an unknown input  $\mathbf{X}$  becomes

$$f(\mathbf{X}) = \mathbf{h}(\mathbf{X}) \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (7)$$

Finally, by defining a kernel matrix satisfying Mercer's conditions as

$$\begin{aligned} \boldsymbol{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^T : \Omega_{\text{ELM}a,b} &= h(\mathbf{x}_a) \cdot h(\mathbf{x}_b) \\ &= K(\mathbf{x}_a, \mathbf{x}_b), \end{aligned} \quad (8)$$

(7) becomes

$$f(\mathbf{X}) = \begin{bmatrix} K(\mathbf{X}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{X}, \mathbf{x}_N) \end{bmatrix}^T \left( \frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\text{ELM}} \right)^{-1} \mathbf{T}. \quad (9)$$

In this study, the function  $f(\mathbf{X})$  is the supplementary air-ratio model.

### 3. Supplementary Air-Ratio Model

**3.1. Model Construction.** The objective of the ELM supplementary air-ratio model is to predict the future air-ratio  $y_p$  when the lambda sensor is under failure or during cold start. Previous air-ratio measurement must not be used as the inputs to the model. Three engine parameters related closely to the air-ratio performance were carefully selected as the model inputs: fuel injection time (FI), engine speed (ES), and throttle position (TP). The order of the system dynamics was chosen to be 2 (i.e., second-order system with 2 past time steps), which gives the minimum prediction error [12]. The structure of the supplementary air-ratio model is shown in Figure 1.

To build the supplementary air-ratio model, experimental engine data was used rather than empirical equation data. A Honda DC5 Type-R test car with K20A i-VTEC engine was employed to perform the experiment. A MoTeC M800 programmable electronic control unit (ECU) with factory calibration data was used as a base controller to control the engine. The car was run over a dyno test, and totally 5800 data samples were acquired using a wide-band lambda sensor subject to random throttle positions. The first 3000 data samples were used as the training dataset,  $\mathcal{D}$ , to build the supplementary air-ratio model. The last 2800 data samples were used as the test dataset,  $\mathcal{D}_t$ , to evaluate the generalization of the built supplementary air-ratio model. After training, the output function of the ELM supplementary model for an unseen case can be written in the following form:

$$y_p = \begin{bmatrix} K(\mathbf{X}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{X}, \mathbf{x}_N) \end{bmatrix}^T \left( \frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\text{ELM}} \right)^{-1} \mathbf{T}, \quad (10)$$

where  $\mathbf{x}, \mathbf{T} \in \mathcal{D}$  are the training data,  $y_p$  is the prediction output vector,  $\mathbf{X}$  is the unseen input vector,  $K(\mathbf{x}_a, \mathbf{x}_b) = \exp(-\|\mathbf{x}_a - \mathbf{x}_b\|^2 / 2\sigma^2)$  is the selected radial basis function

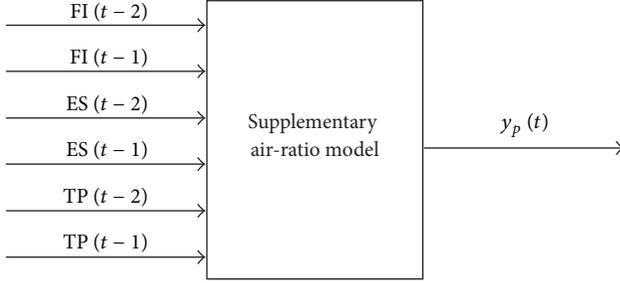


FIGURE 1: Structure of the ELM supplementary air-ratio model.

TABLE 1: Accuracy of different air-ratio models.

Air-ratio model	LMAE
Supplementary air-ratio model	1.8559
Air-ratio model obtained from [12]	2.2424

kernel,  $\mathbf{I}$  is the identity matrix, and  $C$  is the user-specified penalty term. The hyperparameters of  $\sigma^2$  and  $C$  in the kernel-based ELM were tuned by using the hybrid inference introduced in [18], which combined leave-one-out cross validation and Bayesian inference. The details can be found in [18].

**3.2. Evaluation of Engine Air-Ratio Model.** To evaluate the proposed ELM supplementary air-ratio model, its prediction result was compared with the air-ratio model obtained from [12], which is a time-series model with the air-ratio histories as the input. The performance of the models is evaluated in terms of prediction accuracy for unseen case from  $\mathcal{D}_t$ . The prediction results of the two air-ratio models over  $\mathcal{D}_t$  are shown in Figure 2.

From Figure 2, it can be seen that both models achieve comparative performance. To further evaluate the prediction accuracy, the logarithmic mean absolute error (LMAE) for the prediction of each model is determined using the following equation:

$$\text{LMAE} = -\log \left[ \frac{1}{T} \sum_{k=1}^T |y_k - y_p| \right], \quad (11)$$

where  $y_p$  is the model prediction value corresponding to  $\mathbf{X}_k$ ,  $y_k$  is the actual experimental value corresponding to  $\mathbf{X}_k$ , and  $T$  is total number of predictions from  $\mathcal{D}_t$ , which is 2800 in this case study. The larger the LMAE is, the higher the model accuracy is. The evaluation results are provided in Table 1.

The prediction results from Table 1 show that the prediction accuracy of the proposed ELM supplementary air-ratio model is lower than that of the air-ratio model obtained from [12]. It has to be noted that the dynamics of previous air-ratio measurement (i.e.,  $y_p(t-1)$ ,  $y_p(t-2)$ ) are not included in the construction of the supplementary air-ratio model. It is reasonable that the prediction accuracy is not as good as the one using these previous air-ratio histories. However, the accuracy is still acceptable, so ELM was confidently selected to implement the FTC in this study.

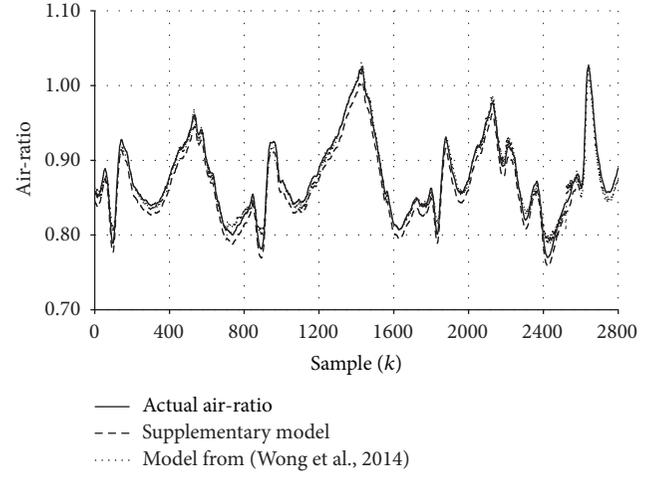


FIGURE 2: Comparison between predicted air-ratios and the corresponding actual air-ratios.

#### 4. MPC Based Fault Tolerance Controller (FTC)

The proposed FTC can be viewed as an improved version of the OEMPC from [12]. Its aim is to maintain the engine air-ratio control performance when the lambda sensor is under failure. The design of the proposed FTC is shown in Figure 3.

The FTC consists of two air-ratio models, including the one from [12] and the supplementary air-ratio model from Section 3, a switch toggling between these two models, and an optimizer based on Brent's method [12, 23]. The engine response over a specified time horizon is predicted by either the air-ratio model from [12] or the supplementary air-ratio model, depending on the condition of the lambda sensor. As far as the lambda sensor works well, the engine response is predicted by the air-ratio model from [12]. Normally, the lambda value for a combustible mixture is in the range of 0.4 to 3 [24]. When the lambda measured is out of this range for a firing engine, it means that the lambda sensor is under failure or during cold start. In this situation, the engine response is predicted by the supplementary air-ratio model. The predictions are used by the optimizer to determine the tentative fuel injection time  $u'$  that minimizes the following performance criterion over the specified time horizon, and then the optimal fuel injection time signal  $u$  is sent to the engine:

$$\begin{aligned} \min J(u') = & \sum_{j=N_1}^{N_2} (y_r(t+j) - y_p(t+j))^2 \\ & + \rho \sum_{j=1}^{N_u} (u'(t+j-1) - u'(t+j-2))^2, \end{aligned} \quad (12)$$

where  $N_1$  and  $N_2$  define the prediction horizon,  $t$  and  $N_u$  are the time step and the control horizon, respectively,  $y_r(t+j)$  is the target air-ratio at the time step  $t+j$ ,  $y_p$  is the predicted air-ratio by either of the two air-ratio models at the time step  $t+j$ , and  $\rho$  is a user-defined weight which penalizes excessive

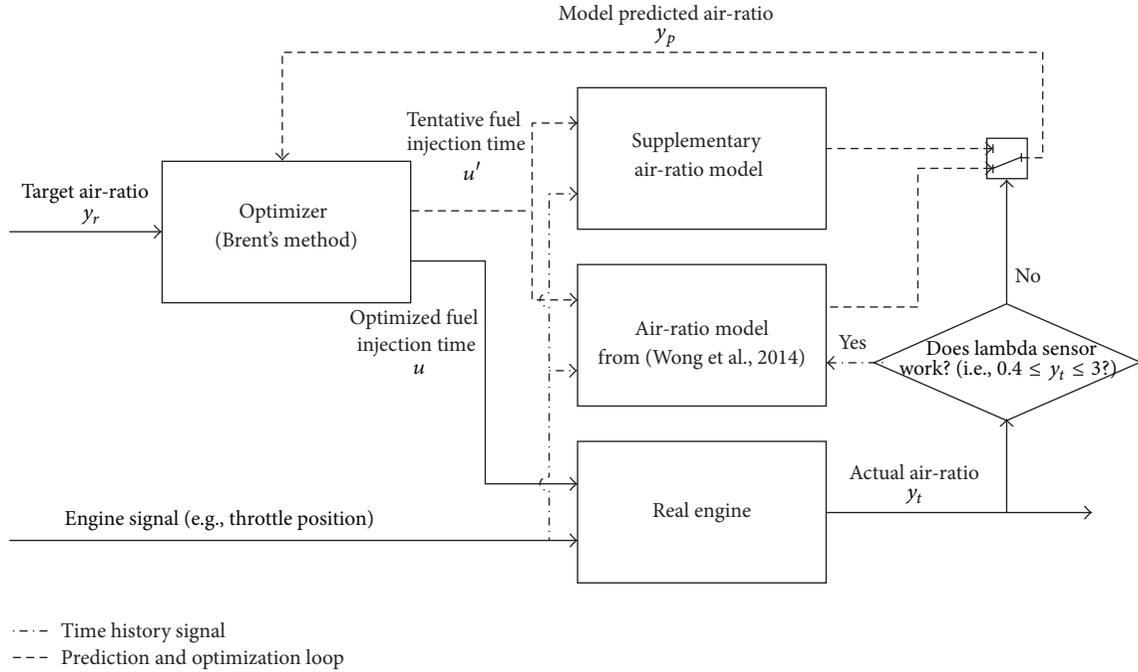


FIGURE 3: Structure of FTC for engine air-ratio control.

movement of the control signal (i.e., the fuel injection time). The variables  $u'(t + j - 1)$  and  $u'(t + j - 2)$  in the second part of (12) are the tentative fuel injection time at the time steps  $t + j - 1$  and  $t + j - 2$ , respectively. The second part of (12) can ensure the stability of the controller output [12]. Moreover, the stability of MPC strategy has also been proved in [25, 26], so the stability of the proposed control scheme could be guaranteed. In short, the proposed control scheme in Figure 3 can provide both regular and fault tolerance control of engine air-ratio.

**4.1. Single Dimension Optimization Approach.** The original optimization problem involved in this paper is multidimensional and constrained with the tentative control signals,  $u'(t), u'(t + 1), \dots, u'(t + N_u - 1)$ , over the control horizon  $N_u$  which can minimize the objective function  $J(u')$  of (12). Then the predicted air-ratios,  $y_p(t + N_1), y_p(t + N_1 + 1), \dots, y_p(t + N_2)$ , can trace the target air-ratios,  $y_r(t + N_1), y_r(t + N_1 + 1), \dots, y_r(t + N_2)$ , by using the optimized fuel injection time-series. Each fuel injection time is normally bounded within the range from 2 ms to 15 ms. However, the multidimensional optimization always requires heavy computation, especially when constraints exist. Real-time control applications often put emphasis on computational speed. The research of [10] showed that the one-dimensional approach is efficient for real-time air-ratio control and the overall tracking error is similar to that using multidimensional optimization approach. Therefore, the optimization problem to be solved is reduced to one dimension. In this paper, the control signal  $u$  is assumed to remain constant over the control horizon. Therefore, the tentative control signal in the objective function is also constant over the control horizon; that is,  $u'(t) = u'(t + 1), \dots, = u'(t + N_u - 1)$ . In this

way, only one parameter  $u'(t)$  is needed to be determined, and the final fuel injection time at each time step  $u$  is set to be the optimal value of  $u'(t)$ .

**4.2. Brent's Method.** There are many optimization techniques available for MPC and each technique has its pros and cons. A well-known technique—Brent's method—was selected as the MPC optimizer in this study for illustrative purpose. Brent's method is a robust and efficient optimization method. It combines the typical parabolic interpolation and golden-section search. The objective function in each iteration is approximated by interpolating a parabola through three existing points. The minimum point of the parabola is taken as a guess for the minimum point if certain criteria are met. Otherwise, golden-section search is carried out. The advantage of this method is that the high convergence rate of parabolic interpolation can be maintained without losing the robustness of golden-section search [23]. The general working principle of Brent's method is shown in Figure 4. The detail optimization procedure of Brent's method was presented in [23] and is not presented herein. There are three parameters of Brent's method, including the initial interval of the input variable,  $[a, b]$  (i.e., the limit of the fuel injection time), the tolerance,  $tol$ , and the maximum iteration for stopping the optimization procedure. The three variables were set to be [2, 15], 0.05, and 50, respectively, because the fuel injection time varies within 2 to 15 ms from 0% to 100% throttle.

## 5. Implementation and Evaluation of FTC

**5.1. Experimental Setup.** To verify its effectiveness, the proposed FTC was implemented and tested on the same test

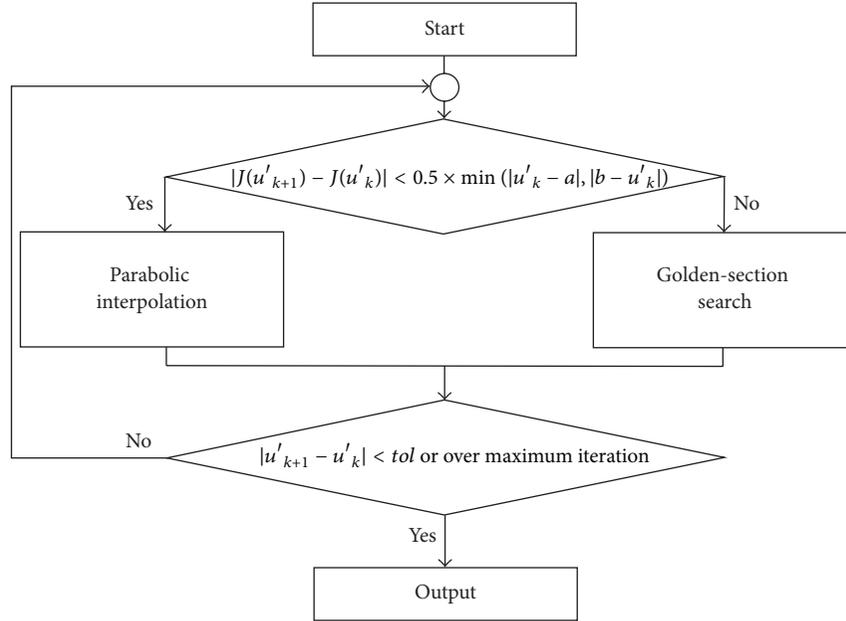


FIGURE 4: General working principle of Brent's method.

car used for sample data collection in Section 3 (Honda DC5 Type-R with K20A i-VTEC engine and MoTeC M800 programmable ECU). The algorithm of FTC was first implemented using MATLAB. A National Instrument (NI) CompactDAQ chassis DAQ-9178 was then employed for signal processing between the MATLAB program and the MoTeC ECU, via NI LabVIEW program. In other words, NI DAQ-9178 serves as an interface between the MATLAB program and the MoTeC ECU. Apart from fuel injector control, the MoTeC ECU also contains many basic control maps, such as ignition map and valve timing map, to maintain the engine operation. The experimental setup and the signal flow between the test car and FTC are shown in Figure 5.

**5.2. Pilot Test-Tracking Ability.** To evaluate the tracking performance of the proposed FTC, a pilot test was carried out. The test was done under the condition that the lambda sensor is already malfunctioning (i.e., the supplementary air-ratio model in the FTC is used). The test cycle for the pilot test is shown in Figure 6, where the throttle position gradually changes from 15% to 75% throttle (the throttle position increases by 15% every 5 s). Such test cycle is designed by referring to [10] which almost covers the whole operating condition. In this test, the air-ratio is required to track the target air-ratios from the stoichiometric value (1.00) for minimum emissions to a value for the best brake-specific fuel consumption (1.05) and then to a value for maximum engine power (0.95) as the throttle position is gradually changed from 15% to 75% throttle. Such tracking of air-ratio changes is essential for automobiles to satisfy the emission, fuel consumption, and power requirements under different operating conditions [6].

After choosing the sampling time to be 0.01 s, the tracking ability of the FTC was examined. By trial-and-error, the parameters of the optimizer were chosen as  $N_1 = 1$ ,  $N_2 = 8$ ,  $\rho = 0.75$ , and  $N_u = 5$ . With the test cycle shown in Figure 6 and the parameters chosen, the air-ratio control result and the corresponding fuel injection time of the FTC are shown in Figure 7. It can be seen from Figure 7 that the FTC can regulate the air-ratio to follow the target air-ratios with acceptable deviation, even when the lambda sensor is under failure.

To further verify the result, three tests were done. The first one was conducted under the condition that the lambda sensor did not work, and traditional air-ratio controller (i.e., factory lookup table) was employed. The other two were carried out under the condition that the lambda sensor could work well, and a diagonal recurrent neural-network model predictive controller (DNMPC), which is a recent air-ratio control algorithm applicable when the lambda sensor works normally [10], was employed along with the proposed FTC for comparison purpose. The air-ratio control results and the corresponding fuel injection time of the three tests are shown in Figures 8, 9, and 10, respectively.

When comparing Figure 8 to Figure 7, it is obvious that the proposed FTC outperforms traditional air-ratio control system when the lambda sensor does not work. Furthermore, by comparing Figures 9 and 10, it can be seen that the proposed FTC can perform slightly better than the DNMPC when the lambda sensor works properly. Finally, comparing Figure 10 to Figure 7, it can be learnt that the proposed FTC can achieve better performance when the lambda sensor is under normal condition. This is not surprising because when the lambda sensor works well, the air-ratio model from [12], which has better prediction capability, is utilized, resulting in

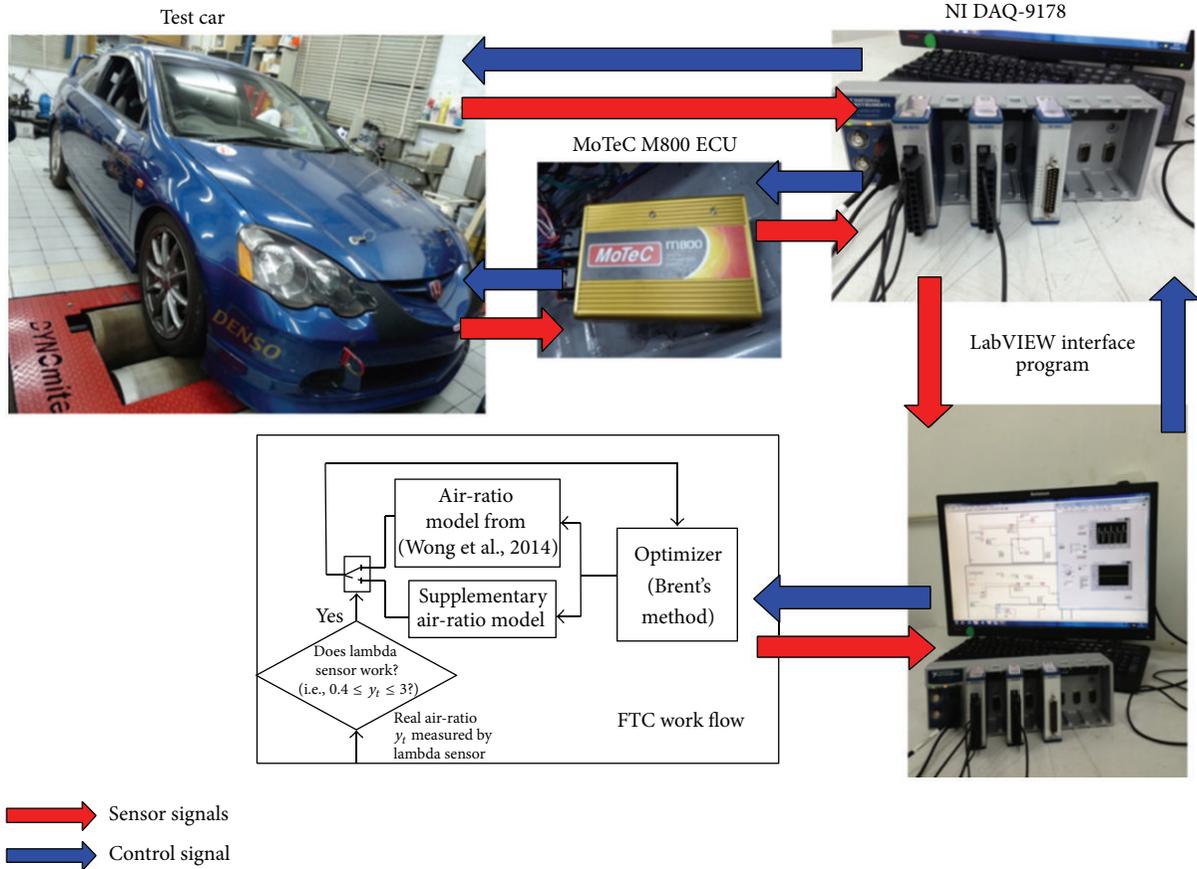


FIGURE 5: Experimental setup and signal flow between test car and FTC.

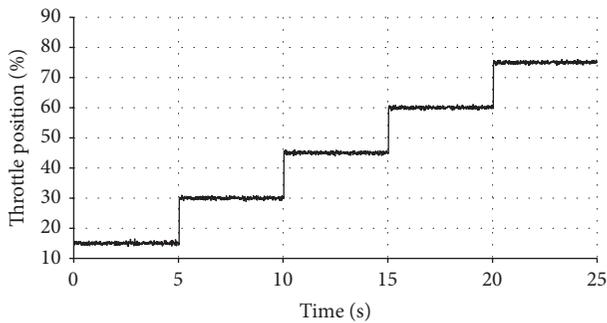


FIGURE 6: Test cycle: throttle position versus time.

better control performance. For comparison purpose, LMAE is again chosen as the tracking index (TI) to evaluate the tracking ability of the controllers, defined by

$$TI = -\log \left[ \frac{1}{T_s} \sum_{k=1}^{T_s} |y_t - y_r(t)| \right], \quad (13)$$

where  $t$  is time step,  $T_s$  is the total number of time step in the test,  $y_t$  is the actual air-ratio at each time step, and  $y_r(t)$  is the corresponding target air-ratio at each time step. The target air-ratio varies according to the aforesaid practical operating

TABLE 2: Performance of different controllers for the pilot test.

Controller	Lambda sensor	TI	Maximum overshoot
FTC	Under failure	2.1022	0.1188
Traditional system	Under failure	1.6244	0.1235
DNMPC	Normal	2.2896	0.1267
FTC	Normal	2.3309	0.1074

conditions. The corresponding results under different lambda sensor conditions are shown in Table 2.

The results in Table 2 show that, even when the lambda sensor is under failure, the control performance of the FTC can still maintain about 90% of that under normal lambda sensor condition. Table 2 also shows that the performance of the proposed FTC under sensor failure is similar to that of DNMPC. The reason may be that the accuracy of the ELM supplementary model is similar to that of the time-series diagonal neural-network model in [10]. Moreover, as compared to traditional air-ratio control system (i.e., factory lookup table), the proposed FTC can improve the air-ratio control performance at about 30% when the lambda sensor is under failure, indicating that the overall air-ratio control performance of the FTC is satisfactory.

Air-ratio model prediction accuracy is one of the most important factors affecting the control performance of

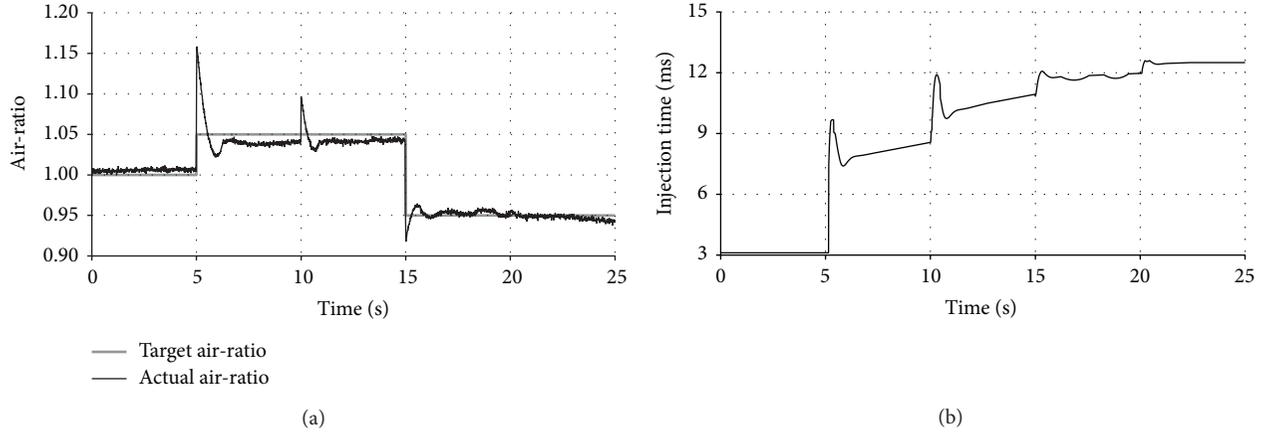


FIGURE 7: (a) Air-ratio control results and (b) corresponding fuel injection time of FTC under lambda sensor failure.

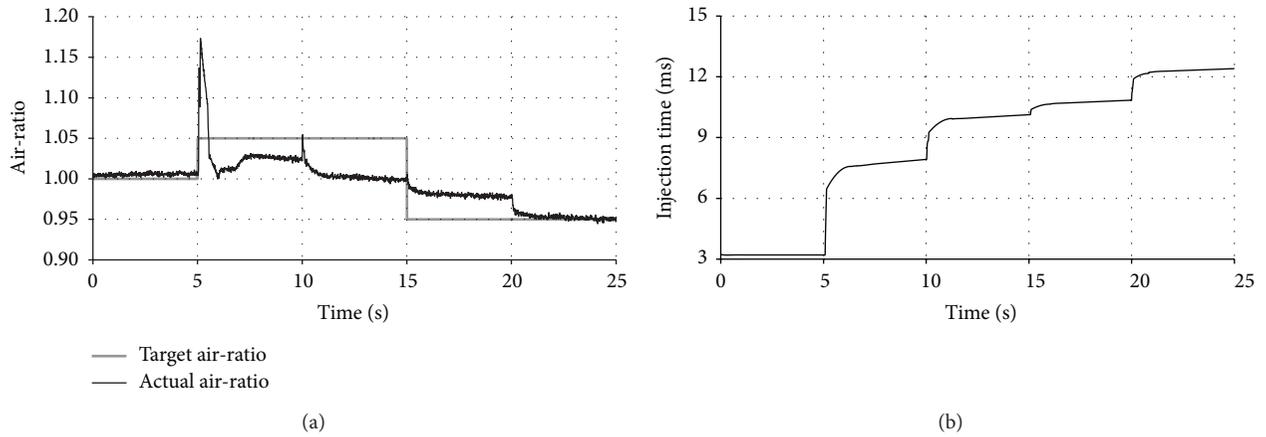


FIGURE 8: (a) Air-ratio control results and (b) corresponding fuel injection time of traditional air-ratio control system under lambda sensor failure (i.e., using factory lookup table only).

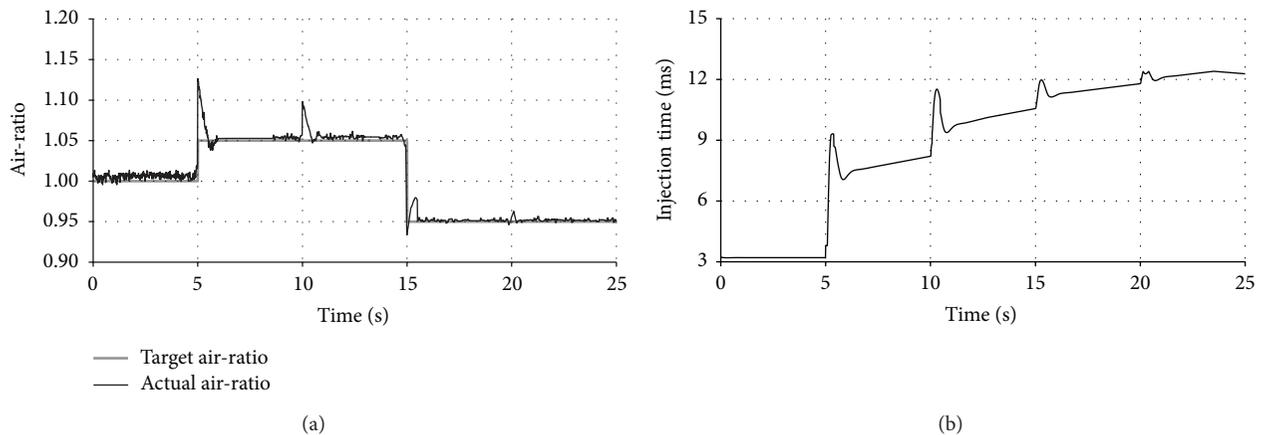


FIGURE 9: (a) Air-ratio control results and (b) corresponding fuel injection time of DN MPC under normal lambda sensor condition.

the model predictive controllers. Therefore, as the air-ratio measurement cannot be provided for model prediction when the lambda sensor is under failure, degrading of the air-ratio control performance is unavoidable. Thankfully, with the

proposed ELM supplementary air-ratio model, the control performance can still be maintained at an acceptable range. This implies that the proposed model and control strategy is feasible.

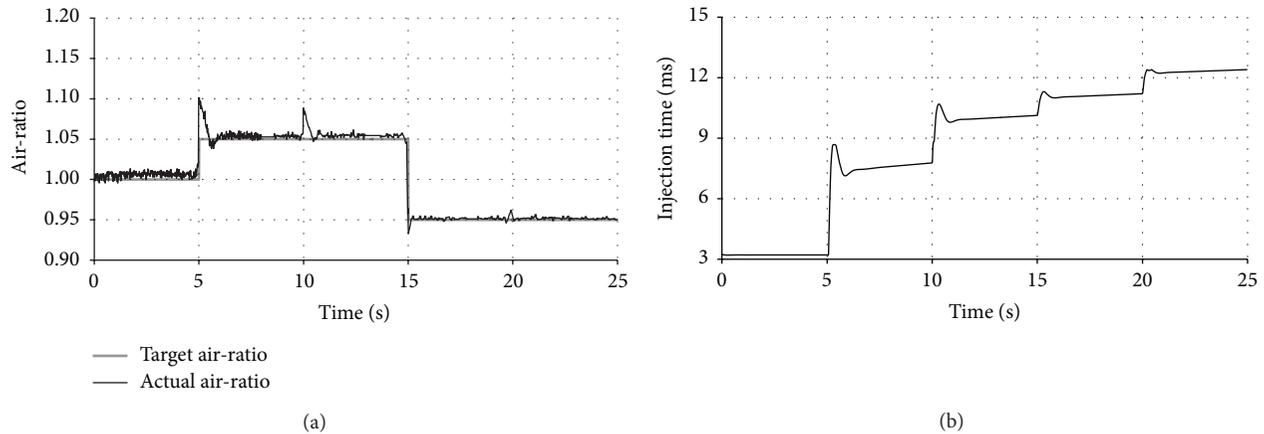


FIGURE 10: (a) Air-ratio control results and (b) corresponding fuel injection time of FTC under normal lambda sensor condition.

## 6. Conclusions

In this study, a supplementary air-ratio model was constructed using kernel-based ELM, in which the air-ratio time histories were not included in the model input. The purpose of the model is to predict the future air-ratio dynamics when the previous air-ratio measurement is unavailable. With the supplementary air-ratio model, a FTC for engine air-ratio control was developed as the first attempt in the literature. In traditional case, the lambda sensor is assumed to work properly, so traditional air-ratio control system may not perform well when the lambda sensor is malfunctioning. The purpose of the proposed FTC is to maintain the air-ratio control performance even when the lambda sensor is under failure or during cold start.

To evaluate its performance, the proposed FTC was successfully implemented and tested on a real automotive engine. Experimental results show that when the lambda sensor does not work well, the air-ratio control performance of the FTC can be effectively maintained over 90% of that under normal lambda sensor condition. Therefore, the FTC is a promising backup engine air-ratio control strategy. In other words, the proposed control scheme in Figure 3 is very suitable to implement in the automotive ECU to provide both regular and fault tolerance control of engine air-ratio.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The research is supported by the University of Macau Research Grant (Grant nos. MYRG2014-00178-FST and MYRG075(Y1-L2)-FST13-VCM). The research is also supported by the Science and Technology Development Fund of Macau (Grant no. FDCT/075/2013/A).

## References

- [1] J. Heinrich, P. E. Schwarze, N. Stilianakis et al., *Health Effects of Transport-Related Air Pollution*, World Health Organization, 2005.
- [2] S. S. Lim, T. Vos, A. D. Flaxman et al., "A comparative risk assessment of burden of disease and injury attributable to 67 risk factors and risk factor clusters in 21 regions, 1990–2010: a systematic analysis for the Global Burden of Disease Study 2010," *The Lancet*, vol. 380, no. 9859, pp. 2224–2260, 2010.
- [3] S. H. L. Yim and S. R. H. Barrett, "Public health impacts of combustion emissions in the United Kingdom," *Environmental Science & Technology*, vol. 46, no. 8, pp. 4291–4296, 2012.
- [4] K. Zhang and S. Batterman, "Air pollution and health risks due to vehicle traffic," *Science of the Total Environment*, vol. 450–451, pp. 307–316, 2013.
- [5] J. B. Heywood, *Internal Combustion Engine Fundamentals*, McGraw-Hill, New York, NY, USA, 1988.
- [6] T. Gilles, *Automotive Service: Inspection, Maintenance, Repair*, Cengage Learning, 2011.
- [7] S. Pace and G. G. Zhu, "Sliding mode control of both air-to-fuel and fuel ratios for a dual-fuel internal combustion engine," *Journal of Dynamic Systems, Measurement and Control*, vol. 134, no. 3, Article ID 031012, 2012.
- [8] R. Tafreshi, B. Ebrahimi, J. Mohammadpour, M. A. Franchek, K. Grigoriadis, and H. Masudi, "Linear dynamic parameter-varying sliding manifold for air-fuel ratio control in lean-burn engines," *IET Control Theory & Applications*, vol. 7, no. 10, pp. 1319–1329, 2013.
- [9] B. Ebrahimi, R. Tafreshi, H. Masudi, M. Franchek, J. Mohammadpour, and K. Grigoriadis, "A parameter-varying filtered PID strategy for air-fuel ratio control of spark ignition engines," *Control Engineering Practice*, vol. 20, no. 8, pp. 805–815, 2012.
- [10] Y.-J. Zhai, D.-W. Yu, H.-Y. Guo, and D. L. Yu, "Robust air/fuel ratio control with adaptive DRNN model and AD tuning," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 2, pp. 283–289, 2010.
- [11] T. Sardarmehni, J. Keighobadi, M. B. Menhaj, and H. Rahmani, "Robust predictive control of lambda in internal combustion engines using neural networks," *Archives of Civil and Mechanical Engineering*, vol. 13, no. 4, pp. 432–443, 2013.

- [12] P. K. Wong, H. C. Wong, C. M. Vong, Z. Xie, and S. Huang, "Model predictive engine air-ratio control using online sequential extreme learning machine," *Neural Computing and Applications*, 2014.
- [13] P. K. Wong, L. M. Tam, K. Li, and C. M. Vong, "Engine idle-speed system modelling and control optimization using artificial intelligence," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 224, no. 1, pp. 55–72, 2010.
- [14] Y.-W. Kim, G. Rizzoni, and V. I. Utkin, "Developing a fault tolerant power-train control system by integrating design of control and diagnostics," *International Journal of Robust and Nonlinear Control*, vol. 11, no. 11, pp. 1095–1114, 2001.
- [15] E. Gassenfeit and J. Powell, "Algorithms for air-fuel ratio estimation using internal combustion engine cylinder pressure," SAE Technical Paper 890300, 1989.
- [16] J. R. Asik, G. M. Meyer, and D. X. Tang, "A/F ratio estimation and control based on induced engine roughness," *IEEE Control Systems Magazine*, vol. 16, no. 6, pp. 35–42, 1996.
- [17] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.
- [18] K. I. Wong, P. K. Wong, C. S. Cheung, and C. M. Vong, "Modelling of diesel engine performance using advanced machine learning methods under scarce and exponential data set," *Applied Soft Computing*, vol. 13, no. 11, pp. 4428–4441, 2013.
- [19] J. W. Cao, Z. P. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [20] I. Lou, Z. Xie, W. K. Ung, and K. M. Mok, "Freshwater algal bloom prediction by extreme learning machine in Macau Storage Reservoirs," *Neural Computing and Applications*, 2014.
- [21] J. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [22] P. K. Wong, K. I. Wong, C. M. Vong, and C. S. Cheung, "Modeling and optimization of biodiesel engine performance using kernel-based extreme learning machine and cuckoo search," *Renewable Energy*, vol. 74, pp. 640–647, 2015.
- [23] T. R. Chandrupatla, "An efficient quadratic fit—sectioning algorithm for minimization without derivatives," *Computer Methods in Applied Mechanics and Engineering*, vol. 152, no. 1–2, pp. 211–217, 1998.
- [24] W. W. Pulkrabek, *Engineering Fundamentals of the Internal Combustion Engine*, Pearson Prentice Hall, 2nd edition, 2004.
- [25] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
- [26] G. Y. Li, *Application of Intelligent Control and MATLAB to Electronically Controlled Engines*, Publishing House of Electronics Industry, Beijing, China, 2007.

## Research Article

# Modeling, Prediction, and Control of Heating Temperature for Tube Billet

Yachun Mao,<sup>1</sup> Dong Xiao,<sup>2</sup> and Dapeng Niu<sup>2</sup>

<sup>1</sup>College of Resources and Civil Engineering, Northeastern University, Shenyang 110004, China

<sup>2</sup>Information Science and Engineering School, Northeastern University, Shenyang 110004, China

Correspondence should be addressed to Dong Xiao; [xiaodong@ise.neu.edu.cn](mailto:xiaodong@ise.neu.edu.cn)

Received 26 May 2014; Revised 4 September 2014; Accepted 7 October 2014

Academic Editor: Yi Jin

Copyright © 2015 Yachun Mao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Annular furnaces have multivariate, nonlinear, large time lag, and cross coupling characteristics. The prediction and control of the exit temperature of a tube billet are important but difficult. We establish a prediction model for the final temperature of a tube billet through OS-ELM-DRPLS method. We address the complex production characteristics, integrate the advantages of PLS and ELM algorithms in establishing linear and nonlinear models, and consider model update and data lag. Based on the proposed model, we design a prediction control algorithm for tube billet temperature. The algorithm is validated using the practical production data of Baosteel Co., Ltd. Results show that the model achieves the precision required in industrial applications. The temperature of the tube billet can be controlled within the required temperature range through compensation control method.

## 1. Introduction

A seamless tube should be heated to a given temperature in an annular furnace prior to piercing. The heating quality of the tube billet directly influences the quality of the seamless tube. In seamless tube production, the main evaluation criterion for tube billet heating quality is the final exit temperature of the tube billet. The heating reaction mechanism of an annular furnace is complicated. It has nonlinear, large time lag, and uncertainty characteristics. The temperature distribution on the surface of a tube billet within the furnace cannot be directly measured and controlled [1]. Thus, controlling the outlet temperature of the billet is difficult.

Measuring the temperature of the billet in the furnace is also difficult. A common method is to establish a temperature prediction model for the billet. Several scholars have proposed mechanism models for billet temperature in furnaces. Chen et al. [2] established a temperature drop model of a tube billet during the transfer of the tube billet from the heating furnace to the rolling mill based on the partial differential equation of heating transfer. Jaklič et al. [3] established a mathematical model of deformation and heat flow during rough rolling. Considering that mechanism modeling is

complex and modeling consumes much time, a model is usually established with a single furnace type and under many restricting conditions. Zhang et al. [4, 5] estimated the tube billet exit temperature through dynamic modeling of furnace temperature; however, the prediction error was too large to meet the requirements of the heating process. Wick [6] applied Kalman filter technique to estimate the temperature distribution of a tube billet inside a heating furnace. The disadvantage of this method is that the surface temperature of the tube billet inside the furnace must be measured, which is difficult to perform in practical production. Xiao et al. [7] employed production data and applied PCR method to establish a soft measurement model of tube billet temperature inside a furnace. However, this model has poor prediction precision because of the strong nonlinearity of production data. Chen and Chai [8] designed a preprocessing system for production process data. This system can predict several variables that are difficult to measure through the use of a self-adapting fuzzy-neural network. Cui and Ding [9] established a soft measurement model of tube billet temperature based on RBF neural network; however, model update was not considered. Iwamoto et al. [10] designed an automatic control system for a tube billet reheating rotary hearth furnace.

The system consists of a component that calculates the tube billet temperature and a component that calculates the optimal furnace temperature set point. In large Chinese steel companies, such as Baosteel Co., Panzhihua Iron and Steel Co., Ltd., Anshan Iron and Steel Co., Ltd., and Capital Iron and Steel Co., Ltd., temperature prediction models are utilized in several heating furnaces. However, their prediction models are almost entirely engineering models imported from abroad. Therefore, these models are difficult to maintain and transplant, and the costs of doing so are high.

With the development of configuration software and database technique, increasing amounts of production data are being collected and stored. Therefore, data-driven modeling and control methods are eliciting more and more attention. He et al. [11] and Lv et al. [12] established data models for the Ladle furnace through data-driven methods. In the present work, the production data of an annular furnace were obtained from the seamless tube subcompany of Baosteel. Industrial process data contain noise, which reduces the modeling accuracy of the extreme learning machine (ELM) algorithm. By contrast, the capability of the partial least square (PLS) algorithm to process linear relevant data is suitable. Moreover, some cyclical changes occur in production. Thus, the model should be updated in real time. Online sequential (OS) ELM and recursive PLS algorithm can realize model update. Online sequential extreme learning machine dynamic recursive partial least square (OS-ELM-DRPLS) algorithm was proposed in this study. A tube billet temperature prediction model based on this algorithm was established, and a strategy for the optimization and control of tube billet temperature was proposed based on this model. OS-ELM-DRPLS not only has the advantages of the OS-ELM algorithm (e.g., rapid nonlinear modeling and update) but also has the capability of the RPLS algorithm to process linear relevant data. The large time lag and reduced model precision were solved through dynamic processing. The algorithm is easy to implement with advanced computer language. Configuration software, such as WINCC6.1, can be utilized to compile the modeling and control algorithms into special modules for use in industrial sites. Simulation and actual experiments prove that tube billet temperature can be predicted and controlled within the scope of the production process requirements with the established temperature prediction model and the proposed strategy of optimization and control based on OS-ELM-DRPLS algorithm.

## 2. Annular Furnace

An annular furnace is a type of rotary hearth furnace utilized for tube billet heating. It consists of the furnace shaft and auxiliary equipment for charging and discharging. The furnace shaft consists of a fixed furnace roof, a ring-type tunnel surrounded by a fixed furnace roof wall, and a circular ring-like rotary hearth, as shown in Figure 1.

External and inner ring seams are placed between the fixed furnace wall and rotary hearth in an annular furnace. Internal and external water seal tanks are arranged

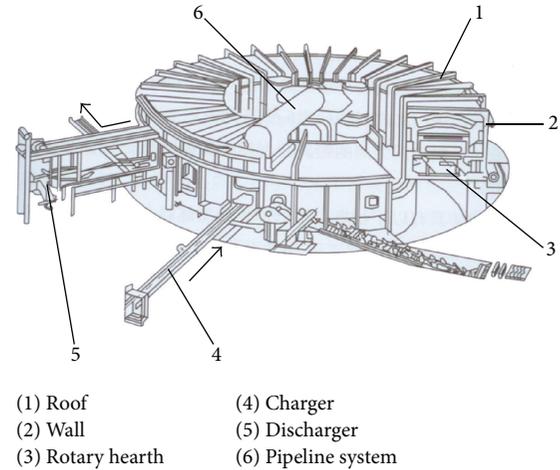


FIGURE 1: Schematic of annular furnace.

beneath the external and inner ring seams to maintain normal temperature and pressure in the furnace cavity and prevent external cold air from entering the furnace cavity. Fuel gas and combustion-supporting air are blown into the furnace through burning nozzles mounted on the external and internal walls or furnace roof to make the gas burn within the furnace and heat the tube billet. The fumes produced by gas burning within the furnace move conversely through the rotary hearth to the tail end of the soaking zone, enter the flue and chimney outside the furnace, and exit to the atmosphere. The external wall of the furnace has charging and discharging furnace doors, in which a charger and discharger are placed, respectively. Charging and discharging proceed simultaneously. When a tube billet is placed in the furnace, the bottom rotates at a certain angle. Tube billets follow a radial layout inside the furnace and are arranged either in a single row or in multirows.

The furnace cavity is divided into preheating, heating, and soaking zones according to the heating process of the tube billet in the annular furnace. Burning nozzles are not mounted in the preheating zone. A flue opening is arranged on the side wall near the charging furnace door in this zone. High-temperature exhaust gas flows toward the opposite direction of hearth rotation and exits into the atmosphere through the flue opening in the heating and soaking zones. During the flow process of high-temperature exhaust gas, the tube billets in the preheating zone are mainly convection heated. The length of the preheating zone accounts for approximately one-fourth of the peripheral length of the annular furnace. Temperature differences between the surface and center and between both ends exist in the tube billet rapidly heated in the heating zone. To reduce the temperature difference of the tube billets and eliminate their male-female faces, the tube billets must be heated in the soaking zone. The length of the soaking zone is approximately three-twentieths of the peripheral length of the annular furnace. In addition, no tube billet and burning nozzle are present between the charging and discharging furnace doors. A partition wall is placed in the middle. The distance between the charging and

discharging furnace doors is approximately one-tenth of the peripheral length of the annular furnace.

### 3. Dynamic Nonlinear PLS Method

Given that a linear PLS model cannot correctly describe the nonlinear relationship between independent variable  $X$  and dependent variable  $Y$  ( $X$  is the variable matrix that affects the heating temperature of the tube billet and  $Y$  is the variable matrix of the heating temperature), nonlinear PLS (NLPLS) method is required. Wold et al. extended the PLS method to the nonlinear field [13, 14]. Two feasible methods exist in nonlinear PLS methods. One method is to perform array extension for the input matrix, introduce several nonlinear terms of the original variable (e.g., the square term), and then regress the extended input and output matrix through PLS method. If prior knowledge on the relationship of original input variable does not exist, this method cannot be utilized as a reference in the selection of the combined mode and may lead to an oversized dimension of the input matrix and difficulties in processing. The other method is to reserve the linear external model of the PLS method. The internal model is nonlinear. The effect of various input variables on the final tube billet temperature has a different time lag because of the large time lag characteristics of tube billet heating. Accurately predicting the tube billet temperature through traditional PLS method is difficult. In this study, dynamic PLS method was utilized to calculate the lag time of various input channels and significantly improve the model's precision. The algorithm is as follows:

$$X = [x_1^{K1}, x_2^{K2}, \dots, x_p^{Kp}], \quad (1)$$

where  $K1, K2, \dots, Kp$  are the ratio of lag time to sampling period for sampling variables  $x_1, x_2, \dots, x_p$ .

(1) The external relation model is

$$X = TP^T + E = \sum_{a=1}^A t_a p_a^T + E, \quad (2)$$

$$Y = UQ^T + F = \sum_{a=1}^A u_a q_a^T + F,$$

where  $A$  is the number of reserved eigenvector;  $t_a$  ( $n \times 1$ ) and  $u_a$  ( $n \times 1$ ) are the score vectors of  $X$  and  $Y$ , respectively;  $p_a$  ( $m \times 1$ ) and  $q_a$  ( $p \times 1$ ) are the load vectors of  $X$  and  $Y$ , respectively;  $T$  ( $n \times A$ ) and  $U$  ( $n \times A$ ) are the score matrixes of  $X$  and  $Y$ , respectively;  $P$  ( $m \times A$ ) and  $Q$  ( $p \times A$ ) are the load matrixes of  $X$  and  $Y$ , respectively; and  $E$  and  $F$  are the fit residual matrixes of  $X$  and  $Y$ , respectively.

(2) The internal relation model is

$$\hat{u}_a = f(t_a) + \varepsilon, \quad (3)$$

where  $f()$  is the nonlinear function and  $\varepsilon$  is the residual.

Given that a neural network is capable of nonlinearity fitting during the modeling of the batch process, nonlinear MPLS method, where the internal model adopts a neural network, has gained extensive applications. Considering that

a traditional feed-forward neural network adopts a gradient learning algorithm during training, the parameters in the network need iteration and updating. Training not only consumes much time but also easily results in issues of local minimum and excessive training [15].

### 4. OS-ELM-DRPLS Algorithm

*4.1. ELM Algorithm.* In supervised batch learning, the learning algorithms employ a finite number of input-output samples for training [16–22]. For  $N$  arbitrary distinct samples  $(x_i, t_i) \in R^n \times R^m$ , where  $x_i$  is a  $n \times 1$  input vector and  $t_i$  is a  $m \times 1$  target vector, if a single hidden layer feed-forward neural network (SLFN) [23–26] with  $\tilde{N}$  hidden nodes can approximate these  $N$  samples with zero error, then  $\beta_i, a_i$ , and  $b_i$  exist such that

$$f_{\tilde{N}}(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(a_i, b_i, x_j) + \varepsilon_j = t_j. \quad (4)$$

In the expression above,  $j = 1, \dots, N$ ,  $a_i$ , and  $b_i$  are the learning parameters of the hidden nodes (the weight vector connecting the input node to the hidden node and the threshold of the hidden node) randomly selected according to the proof provided by Huang et al.  $\beta_i$  is the weight connecting the  $i$ th hidden node to the output node. Error term  $\varepsilon_j$  is added to avoid overfitting the noise in the data.  $G(a_i, b_i, x)$  is the output of the  $i$ th hidden node with respect to input  $x$ , and  $\tilde{N}$  is the number of hidden nodes that can be determined by trial and error or prior experience. Equation (4) can then be written compactly as

$$H\beta = T, \quad (5)$$

where

$$H(a_1, \dots, a_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, x_1, \dots, x_N) = \begin{bmatrix} G(a_1, b_1, x_1) & \dots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_1) \\ \vdots & \dots & \vdots \\ G(a_1, b_1, x_N) & \dots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_N) \end{bmatrix}_{N \times \tilde{N}}, \quad (6)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}.$$

In the expressions above,  $H$  is the hidden layer output matrix of the network; the  $i$ th column of  $H$  is the  $i$ th hidden node's output vector with respect to inputs  $x_1, x_2, \dots, x_N$ , and the  $j$ th row of  $H$  is the output vector of the hidden layer with respect to input  $x_j$ . The hidden node parameters  $a_i$  and  $b_i$  need not be tuned during training and may simply be assigned with random values. Equation (5) then becomes a linear system, and the output weights  $\beta$  are estimated as

$$\tilde{\beta} = H^+ T, \quad (7)$$

where  $H^+$  is the Moore-Penrose generalized inverse of hidden layer output matrix  $H$  [27, 28].

4.2. *OS-ELM Algorithm.* In actual applications, training data may arrive chunk by chunk or one by one. Hence, the batch ELM algorithm has to be modified and made online sequential for this case [29, 30].

Output weight matrix  $\hat{\beta}$  ( $\hat{\beta} = H^+T$ ) provided in (7) is a least-squares solution of (5). We consider the case, where  $\text{rank}(H) = \tilde{N}$  is the number of hidden nodes. Under this condition,  $H^+$  of (7) is provided by

$$H^+ = (H^T H)^{-1} H^T. \quad (8)$$

If  $H^T H$  is singular, one can make it nonsingular by selecting a small network size  $\tilde{N}$  or increasing data number  $N$  in the initialization phase of OS-ELM. Substituting (8) to (7),  $\hat{\beta}$  becomes

$$\hat{\beta} = (H^T H)^{-1} H^T T. \quad (9)$$

Equation (9) is the least-squares solution to  $H\beta = T$ . Sequential implementation of (9) results in OS-ELM [31].

Given a chunk of initial training set  $\aleph_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$  and  $N_0 \geq \tilde{N}$ , if the batch ELM algorithm is employed, the solution of minimizing  $\|H_0\beta - T\|$ , which is given by  $\beta_0 = K_0^{-1} H_0^T T_0$ , where  $K_0 = H_0^T H_0$ , must be considered.

We consider another chunk of data  $\aleph_1 = \{(x_i, t_i)\}_{i=N_0+1}^{N_0+N_1}$ , where  $N_1$  is the number of samples in this chunk. The problem involves minimizing

$$\left\| \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \beta - \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} \right\|. \quad (10)$$

Considering both  $\aleph_0$  and  $\aleph_1$ , output weight  $\beta$  becomes

$$\beta_1 = K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} \quad \text{where } K_1 = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}. \quad (11)$$

For sequential learning,  $\beta_1$  should be expressed as a function of  $\beta_0$ ,  $K_1$ ,  $H_1$ , and  $T_1$  and not as a function of dataset  $\aleph_0$ .  $K_1$  can be written as

$$K_1 = \begin{bmatrix} H_0^T & H_1^T \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} = K_0 + H_1^T H_1, \quad (12)$$

$$\begin{aligned} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} &= H_0^T T_0 + H_1^T T_1 = K_0 K_0^{-1} H_0^T T_0 + H_1^T T_1 \\ &= K_0 \beta_0 + H_1^T T_1 = (K_1 - H_1^T H_1) \beta_0 + H_1^T T_1 \\ &= K_1 \beta_0 - H_1^T H_1 \beta_0 + H_1^T T_1. \end{aligned} \quad (13)$$

Combining (11) and (13),  $\beta_1$  is obtained with

$$\begin{aligned} \beta_1 &= K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^T \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} = K_1^{-1} (K_1 \beta_0 - H_1^T H_1 \beta_0 + H_1^T T_1) \\ &= \beta_0 + K_1^{-1} H_1^T (T_1 - H_1 \beta_0), \end{aligned} \quad (14)$$

where  $K_1 = K_0 + H_1^T H_1$ .

When the  $(k+1)$ th chunk of dataset

$$\aleph_{k+1} = \{(x_i, t_i)\}_{i=(\sum_{j=0}^k N_j)+1}^{\sum_{j=0}^{k+1} N_j} \quad (15)$$

is received, where  $k \geq 0$  and  $N_{k+1}$  denotes the number of samples in the  $(k+1)$ th chunk, we have

$$K_{k+1} = K_k + H_{k+1}^T H_{k+1}, \quad (16)$$

$$\beta_{k+1} = \beta_k + K_{k+1}^{-1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta_k).$$

$K_{k+1}^{-1}$  rather than  $K_k$  is utilized to compute  $\beta_{k+1}$  from  $\beta_k$  in (16). The update formula for  $K_{k+1}^{-1}$  is derived with the Woodbury formula

$$\begin{aligned} K_{k+1}^{-1} &= (K_k + H_{k+1}^T H_{k+1})^{-1} \\ &= K_k^{-1} - K_k^{-1} H_{k+1}^T (I + H_{k+1} K_k^{-1} H_{k+1}^T)^{-1} \times H_{k+1} K_k^{-1}. \end{aligned} \quad (17)$$

We let  $P_{k+1} = K_{k+1}^{-1}$ . The equation for updating  $\beta_{k+1}$  can be written as

$$P_{k+1} = P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k, \quad (18)$$

$$\beta_{k+1} = \beta_k + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta_k).$$

Equation (18) provides the recursive formula for  $\beta_{k+1}$ .

4.3. *OS-ELM-DRPLS Modeling Steps.* The difference of nonlinear DRPLS modeling method based on OS-ELM from linear PLS method is that the former employs ELM to establish the internal nonlinear model and updates the internal and external models. This method reserves the linear external model, extracts the attributive information of the process through PLS, eliminates the colinearity of data, reduces the dimension of the input variable, and then adopts ELM to establish a nonlinear internal model between the input score vector matrix and the output score vector; the nonlinear processing capability of the internal model is enhanced. Thus, OS-ELM-DRPLS method has the advantages of PLS and ELM (i.e., the robustness and feature extraction capability of PLS method and quick nonlinear processing capability of ELM as well as precision accuracy through real-time model update).

The modeling and testing steps of nonlinear DRPLS method based on OS-ELM are as follows.

(1) Two standardized data matrices,  $X \in R^{n \times m}$  and  $Y \in R^{n \times p}$ , are assigned. The dynamic nonlinear PLS regression model can be expressed as follows:

$$X = [x_1^{K1}, x_2^{K2}, \dots, x_p^{Kp}], \quad (19)$$

where  $K1, K2, \dots, Kp$  are the ratio of lag time to the sampling period for sampling variables  $x_1, x_2, \dots, x_p$ .

(2) The batch data of the batch process are deployed, cross-validation method is implemented to determine the number of latent variables, and linear PLS method is applied to calculate score vector matrices  $T$  and  $U$  and load vector matrices  $P$  and  $Q$  for modeling samples  $X$  and  $Y$ :

$$X = TP^T + E = \sum_{a=1}^A t_a p_a^T + E, \quad (20)$$

$$Y = UQ^T + F = \sum_{a=1}^A u_a q_a^T + F.$$

(3) A node number is assigned to the ELM hidden layer and activation function (e.g., sigmoid function), ELM is employed to establish a nonlinear model between internal models  $T$  and  $U$ , and  $U = f_{\text{ELM}}(T)$  is obtained, where  $f_{\text{ELM}}(\cdot)$  is the nonlinear function indicated by ELM. The hidden nodes in SLFN transform the feature space into another feature space. The original ELM regards the number of nodes as a parameter to be defined. We increase the number of hidden nodes until stop criteria (e.g., residual error reduction) are reached. Meanwhile, the number of hidden nodes is less than  $N$ .

(4) When one new batch of data  $X_1, Y_1$  is obtained, PLS decomposition is performed, and score and load vectors  $T_1, U_1, P_1, Q_1$  are obtained:

$$\begin{aligned} X_1 &= T_1 P_1^T + E, \\ Y_1 &= U_1 Q_1^T + F. \end{aligned} \quad (21)$$

According to (18), the OS-ELM algorithm is adopted to update the output layer weight value and the internal model. Weighted mean is conducted on the load matrix of the external model, and external RPLS update, where  $w$  is the weight value factor, is achieved. The above steps are repeated, and model update is conducted for every batch:

$$\begin{aligned} P^T &= wP^T + (1-w)P_1^T, \\ Q^T &= wQ^T + (1-w)Q_1^T. \end{aligned} \quad (22)$$

(5) Testing data are utilized to verify the model's precision. PLS decomposition is conducted on testing data  $X_2$ , and score vector  $T_2$  is obtained:

$$X_2 = T_2 P^T + E. \quad (23)$$

$T_2$  is introduced into the OS-ELM model.  $U_2 = f_{\text{OS-ELM}}(T_2)$  is obtained, and the model prediction value is determined through  $\hat{Y} = UQ^T$ .

(6) A system error is obtained by comparing  $\hat{Y}$  with the practical output.  $K1, K2, \dots, Kp$  can vary within  $1 - n$ . After each variation,  $x_1^{K1}, x_2^{K2}, \dots, x_p^{Kp}$  are substituted back to (19) for calculation and to obtain an estimation error. Finally, one is obtained by exhausting a group of optimal  $K1, K2, \dots, Kp$  values to minimize the model estimation error:

$$W = \sum_{i=1}^{n/2} |\hat{Y}(i) - Y(i)|. \quad (24)$$

Model parameters  $K1, K2, \dots, Kp$  and  $\beta$  of the OS-ELM-DRPLS model are then obtained through the aforementioned calculation.

## 5. Prediction and Control of Tube Billet Heating Quality Based on OS-ELM-DRPLS Model

*5.1. Introduction of the Site and Selection of Measuring Points.* In the seamless tube subcompany of Baosteel, the designed output of an annular furnace was 160 t/h. Its intermediate diameter was 35 m, and the effective width of hearth was 4.5 m. The hearth was divided into six burning control sections. The diameter of the heated tube billet was 178 mm. The temperature upon entering the furnace was 20°C, and the maximum temperature upon leaving the furnace was 1280°C. In the annular furnace, mixed gas that consists of 52% blast furnace gas, 13% converter gas, and 34.8% coke oven gas was utilized. The composition of the blast furnace gas was 23.5% CO, 2% H<sub>2</sub>, 19.5% CO<sub>2</sub>, and 35% N; the composition of coke oven gas was 53% H<sub>2</sub>, 29.2 CH<sub>4</sub>, 2.8% weight carbon hydride, 7.5% CO, 2.0% CO<sub>2</sub>, 0.6% O<sub>2</sub>, and 4.4% N<sub>2</sub>. The composition of converter gas was 56% CO, 24% N<sub>2</sub>, and 19.7% CO<sub>2</sub>. The specific technical parameters are shown in Table 1.

The final exit temperature of the tube billet was predicted through OS-ELM-DRPLS method. First, the variation in the tube billet temperature was reflected, and the measured variables were easily obtained. On one hand, gas could not be obtained through the peep holes because the peep holes in the furnace were closed. On the other hand, opening of the furnace door to obtain gas affects the testing precision because of the absorption of cold air. Therefore, the measuring points in the site were set at the lighting holes of burning nozzles in the external surrounding furnace walls. Six flow rate detecting points were set for the burning nozzles. Nine thermocouples were mounted in the six working sections to measure the temperature inside the furnace cavity. The specific positions of flow rate and furnace temperature are shown in Figure 2. Fifteen measuring variables were selected to predict the final tube billet exit temperature.  $x_1-x_6$  were measuring points for numbers 1-6 burning nozzle flow rate, and  $x_7-x_{15}$  were measuring points for numbers 1-9 furnace cavity temperature. The variable table is shown in Table 2. After selecting the measuring variables and gathering site

TABLE 1: Main technical parameters of annular furnace.

Furnace output	160 t/h
Specification of tube billet	$\Phi 175$ mm, 860–4500 mm in length, maximum weight per piece 850 kg
Furnace size	Intermediate diameter 35 m, effective width of furnace cavity 5 m, height of furnace cavity 3 m (one section in the preheating zone), 2.5 m (2 sections), 2 m (3–6 sections), total number of hearth batch bins 391, number of tube billets 381 pieces
Calorific value of combustion	Heavy oil 37620 KJ/kg, mixed gas 9196 KJ/m <sup>3</sup>
Demand of combustion	Heavy oil 6755 kg/h, mixed gas 30545 m <sup>3</sup> /h
Arrangement of burning nozzle	Total of 96 side burning nozzles for either oil or gas used in sections 1–6; heat is not provided in the preheating zone
Maximum furnace cavity temperature	Approximately 1400°C
Temperature of tube billet	Enter furnace at 20°C, leave furnace at 1280°C, cross-section temperature difference of leaving furnace $\pm 10^\circ\text{C}$
Charging and discharging rhythm	Maximum 270 piece/h, equivalent to discharging interval of 13.3 s/piece

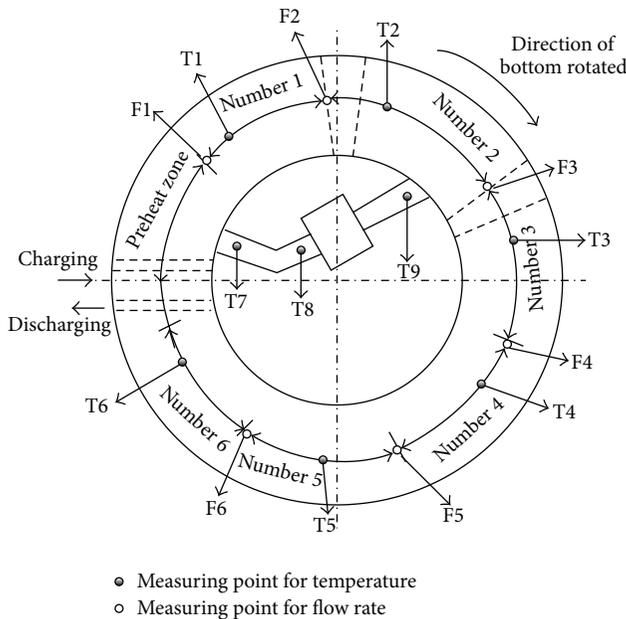


FIGURE 2: Measuring point distribution diagram for the annular furnace.

production data, OS-ELM-DRPLS method was applied to the prediction model of tube billet temperature.

*5.2. Establishment and Checking of the Tube Billet Final Temperature Prediction Model.* The production data for 70 pieces of tube billets produced by Baosteel in March, 2013, were utilized. The first forty samples were utilized as training data to establish the prediction model of tube billet final temperature. The last thirty samples were used for model update. Lump update was employed. Every group of five was considered a lump. The model was updated. The last thirty samples acted as the testing samples to check the precision of model prediction. Prior to modeling, data were expanded, they were standardized-processed, and they underwent cross

TABLE 2: Variables in the modeling of tube billet final temperature.

Ser. number	Variable name	Variable meaning	Unit
1	$x_1$	Number 1 burning nozzle flow rate	m <sup>3</sup> /h
2	$x_2$	Number 2 burning nozzle flow rate	m <sup>3</sup> /h
3	$x_3$	Number 3 burning nozzle flow rate	m <sup>3</sup> /h
4	$x_4$	Number 4 burning nozzle flow rate	m <sup>3</sup> /h
5	$x_5$	Number 5 burning nozzle flow rate	m <sup>3</sup> /h
6	$x_6$	Number 6 burning nozzle flow rate	m <sup>3</sup> /h
7	$x_7$	Number 1 furnace cavity temperature	°C
8	$x_8$	Number 2 furnace cavity temperature	°C
9	$x_9$	Number 3 furnace cavity temperature	°C
10	$x_{10}$	Number 4 furnace cavity temperature	°C
11	$x_{11}$	Number 5 furnace cavity temperature	°C
12	$x_{12}$	Number 6 furnace cavity temperature	°C
13	$x_{13}$	Number 7 furnace cavity temperature	°C
14	$x_{14}$	Number 8 furnace cavity temperature	°C
15	$x_{15}$	Number 9 furnace cavity temperature	°C

checking. The number of PLS potential variables was determined to be 4. The number of ELM hidden layer nodes was 10. The excitation function was a sigmoid function. The ratio

TABLE 3: RMSE and modeling time of different models.

Method	RMSE (test)	Time/s
RPLS	10.2	0.2132
RBF-PLS	4.2	3.0692
OS-ELM-DRPLS	3.1	0.6239

of lag time of  $K_1, K_2, \dots, K_p$  in (19) was calculated in formulas equation (25).

The same data were tested with RPLS, RBF-PLS, and OS-ELM-DRPLS methods. The predicted mean square error and modeling time are shown in Table 3. Although the three methods meet the requirements of industrial application, OS-ELM-RPLS method exhibits better expansion capability, prediction precision, and nonlinear fitting capability for industrial application than RPLS method. Compared with nonlinear RBF-PLS method, the training time in OS-ELM-RPLS method is shorter. OS-ELM-RPLS method can achieve rapid modeling and model update and is significant to the intermittent production processes, such as tube billet heating:

$$\begin{aligned} & [K_1, K_2, \dots, K_{15}] \\ & = [58, 55, 51, 46, 42, 36, 61, 56, 51, 45, 39, 35, 52, 58, 60]. \end{aligned} \quad (25)$$

The unit of  $[K_1, K_2, \dots, K_{15}]$  is the sample time. Figure 3 shows a comparison between regression data and practical modeling data using RPLS and OS-ELM-DRPLS models. The maximum error was  $6.9^\circ\text{C}$  and the mean error was  $2.3^\circ\text{C}$ , which meet the requirements of the production site. To further verify the accuracy of the model, new data were introduced into the model and substituted into the following

equation to obtain estimation value  $\hat{Y}_{\text{new}}$  of the new data. The comparison with  $Y_{\text{new}}$  is shown in Figure 4. The maximum error was  $9.8^\circ\text{C}$  and the mean error was  $3.1^\circ\text{C}$ , which meet the requirements of the production site:

$$\hat{Y}_{\text{new}} = f_{\text{OS-ELM-RDPLS}}(X_{\text{new}}). \quad (26)$$

**5.3. Predicted Control of Tube Billet Final Temperature.** The aforementioned data indicate that tube billet exit temperature often fluctuates in the temperature range of  $1200^\circ\text{C}$  to  $1300^\circ\text{C}$  and often deviates from the ideal piercing temperature ( $1270^\circ\text{C}$ ). Such condition degrades the quality of the tube. The tube billet exit temperature should be controlled within the temperature range of  $1255^\circ\text{C}$  to  $1295^\circ\text{C}$ . The gas flow rate can be adjusted according to the prediction, practical measuring, and target temperatures. Its control period was 1 s. An ELM model predicted controller (EPC) was designed for the annular furnace system with the OS-ELM-DRPLS model predictor (EMP), as shown in Figure 5.

The basic operating principle of predictive control is to generate a sequence of control signals at each sample interval that optimize the control effort to follow the reference trajectory exactly [32, 33]. The ELM model predictive control law was obtained by minimizing the following predictive performance criterion:

$$\begin{aligned} J(k) &= \frac{1}{2} \sum_{p=0}^{N_p} (r(k+p) - \hat{y}(k+p))^2 \\ &= \frac{1}{2} (R(k) - Y(k))^T (R(k) - Y(k)) = \frac{1}{2} E^T(k) E(k), \end{aligned} \quad (27)$$

where

$$\begin{aligned} R(k) &= [r(k) \ r(k+1) \ \dots \ r(k+N_p)]^T, \\ Y(k) &= [\hat{y}(k) \ \hat{y}(k+1) \ \dots \ \hat{y}(k+N_p)]^T, \end{aligned} \quad (28)$$

$$E(k) = [r(k) - \hat{y}(k) \ r(k+1) - \hat{y}(k+1) \ \dots \ r(k+N_p) - \hat{y}(k+N_p)]^T.$$

$N_p$  is the predictive output horizon,  $r(k+p)$  is the input reference signal at discrete time  $k+p$ , and  $\hat{y}(k+p)$  is the  $p$  step-ahead prediction of  $y(k)$ . In general,  $N_p$  is selected to include all responses that are significantly affected by the present control. In this study,  $N_p$  is  $\min(K_1, K_2, \dots, K_{15}) = 35$ .

The control,  $u(k) = [u(k) \ u(k+1) \ \dots \ u(k+N_p)]^T$ , was obtained from the optimization of the cost function (29) based on gradient descent method; that is,

$$\begin{aligned} u(k) &= u(k-1) + \Delta u(k) = u(k-1) + \eta \frac{\partial Y^T(k)}{\partial u(k)} E(k) \\ &= u(k-1) + \eta C^T(k) E(k), \end{aligned} \quad (29)$$

where

$$\begin{aligned} C(k) &= \frac{\partial Y^T(k)}{\partial u(k)} \\ &= \begin{bmatrix} \frac{\partial \hat{y}(k)}{\partial u(k)} & \frac{\partial \hat{y}(k)}{\partial u(k+1)} & \dots & \frac{\partial \hat{y}(k)}{\partial u(k+N_p)} \\ \frac{\partial \hat{y}(k+1)}{\partial u(k)} & \frac{\partial \hat{y}(k+1)}{\partial u(k+1)} & \dots & \frac{\partial \hat{y}(k+1)}{\partial u(k+N_p)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}(k+N_p)}{\partial u(k)} & \frac{\partial \hat{y}(k+N_p)}{\partial u(k+1)} & \dots & \frac{\partial \hat{y}(k+N_p)}{\partial u(k+N_p)} \end{bmatrix}. \end{aligned} \quad (30)$$

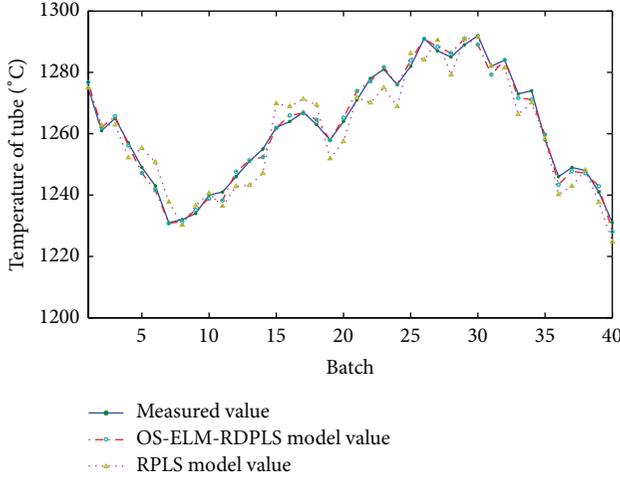


FIGURE 3: Comparison diagram of modeling data.

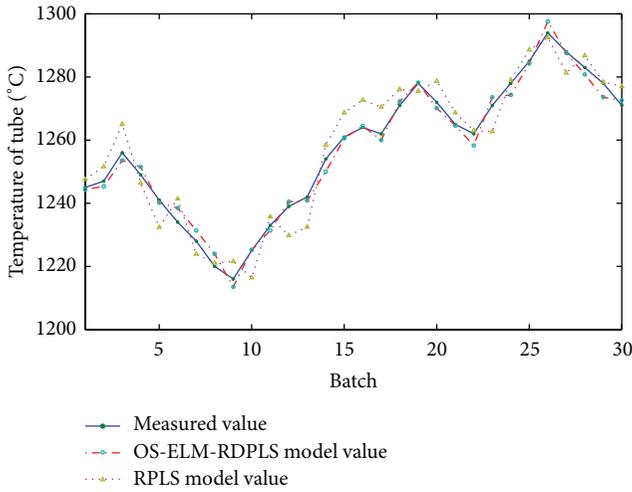


FIGURE 4: Comparison diagram of checking data.

To reduce the computational load of EPC, we let  $u(k + N_p) = \dots = u(k + 1) = u(k)$ . The EPC controller is expressed in the form

$$u(k) = u(k-1) + \eta C^T(k) E(k), \quad (31)$$

where

$$C(k) = \begin{bmatrix} \frac{\partial \hat{y}(k)}{\partial u(k)} & \frac{\partial \hat{y}(k+1)}{\partial u(k)} & \dots & \frac{\partial \hat{y}(k+N_p)}{\partial u(k)} \end{bmatrix}^T. \quad (32)$$

A schematic of the proposed PLC-based temperature control system is shown in Figure 6. The actual temperature control system of the annular furnace is depicted in Figure 7. SIMATIC S7-400 was selected as the PLC of the control system. The entire system is mainly composed of a PLC master station, a remote I/O station, an operator station, a programmer and communication bus, and other components. The main modules of PLC include nine slot bases (UR2), a 4 A power supply module (PS407), a central

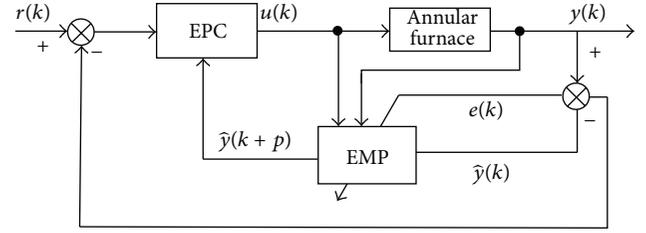


FIGURE 5: Architecture of the annular furnace employing OS-ELM-RDPLS-based predictive control.

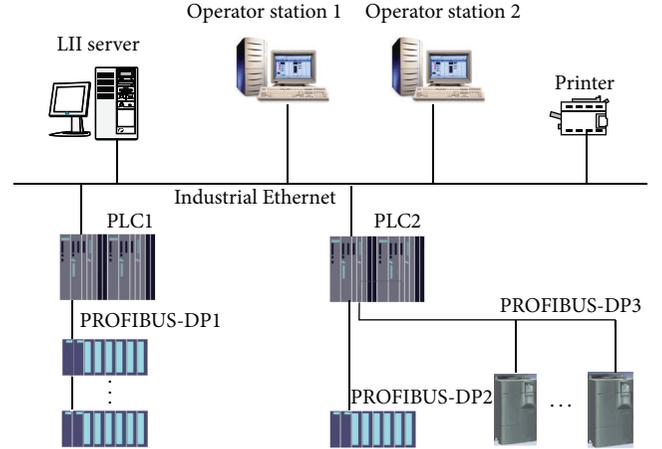


FIGURE 6: Schematic of the PLC-based temperature control system.

processor (CPU416-2DP), 1 M memory card, and a network communication module (CP443-1). The main modules of I/O expansion include a power supply module (PS307), an interface module (IM153-1), a digital input module (SM321 DC24V  $\times$  DI16), a digital output module (SM322 DC24V  $\times$  DO16), a counter function module (8CH FM350-2), an eight-thermocouple input module (SM331), an eight-RTD input module (SM331), and a four-output module (SM332). The main modules of the workstation include a CPU, (Intel Core i7-930, 2.8 GHz  $\times$  4), hard disk (WD 2TB), memory (Kingston 8 GB), color LED (24", 1280  $\times$  1024 resolution), and a net card (Siemens 10/100 MB). The main module of communication includes Ethernet SINEC H1 and field bus PROFIBUS-DP. The main Software programs are Windows 2003 Prof, STEP7 V5.4, and WINCC6.1.

The tube billet exit temperature should be controlled as best as possible within the temperature range of 1255°C to 1295°C. Thirty tube billets were controlled by ELM model predicted control. A thermocouple was "buried" in a tube billet. The temperature course of the tube billet with the buried thermocouple is shown in Figure 8. In position 30, the predicted temperature of the tube billet is 1211. OS-ELM-RDPLS-based predictive control algorithm was employed to make the tube billet reach the lowest required temperature (1255°C). By adjusting the input, the temperature of the tube billet reached 1265°C. The variation in tube billet temperature after introducing temperature compensation control is shown in Figure 9. The variation in tube billet temperature after

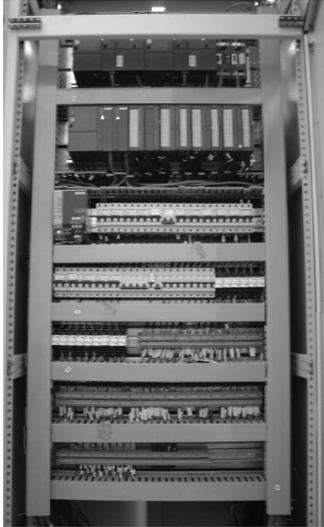


FIGURE 7: Actual temperature control system of the annular furnace.

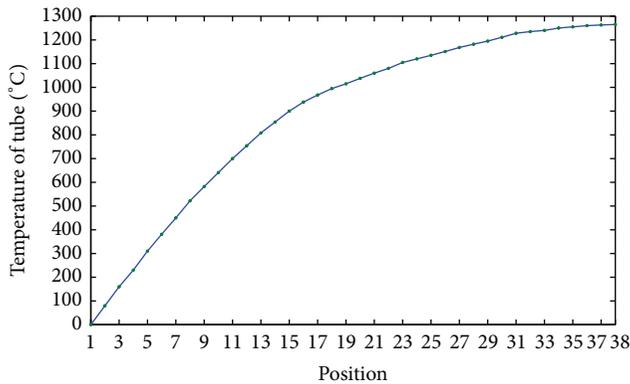


FIGURE 8: Temperature course of the tube billet with a thermocouple.

introducing PID temperature control is shown in Figure 10. The effect of predicted control is better than that of the PID method.

Figure 9 shows that the tube billet exit temperature basically fluctuates in the range of [1255°C, 1295°C]; the tube billet heating quality is better than that before prediction control and meets the requirements of piercing production for tubes.

### 6. Conclusion

Measuring and controlling tube billet heating temperature are difficult because of the complex reaction mechanism during the heating process in an annular furnace. A tube billet final temperature prediction model was established in this study through OS-ELM-DRPLS modeling method. An OS-ELM-DRPLS-based predictive controller for the control of tube billet temperature was also systematically developed. The tube billet heating quality increased to a certain extent. This finding lays the foundation for the improvement of seamless

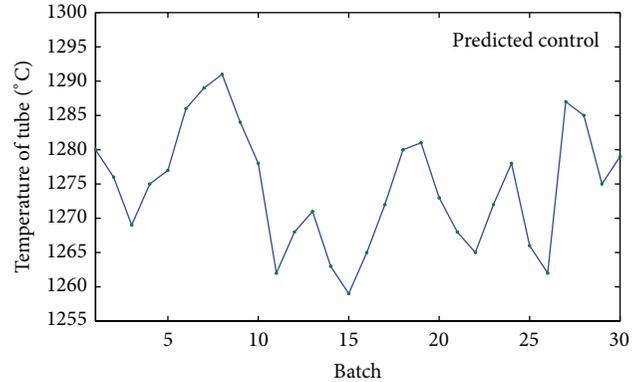


FIGURE 9: Temperature of the tube billet after introducing temperature compensation control.

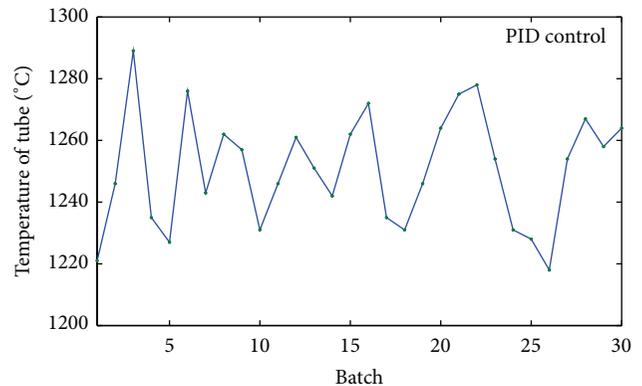


FIGURE 10: Temperature of the tube billet of PID method.

tube quality. After the developed model was compiled into a universal module through the advanced computer language of the configuration software, the modules not only assisted in production by guiding front line workers to operate manually but also formed a perfect close loop control circuit together with the heating furnace model and controller. Hence, tube billet heating quality was improved effectively. Experimentation proved that this method is feasible. This modeling method is also versatile and can be extended to other processes with a large time lag.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant nos. 61203214, 41371437, and 61304121) and Provincial Science and Technology Department of Education Projects, the General Project (L2013101).

## References

- [1] A. D. Acharya and S. Chattopadhyay, "Reheat furnace temperature control and performance at Essar Steel," *Iron and Steel Engineer*, vol. 75, no. 11, pp. 31–36, 1998.
- [2] W. C. Chen, I. V. Samarasekera, A. Kumar, and E. B. Hawbolt, "Mathematical modelling of heat flow and deformation during rough rolling," *Ironmaking and Steelmaking*, vol. 20, no. 2, pp. 113–125, 1993.
- [3] A. Jaklič, B. Glogovac, T. Kolenko, B. Zupančič, and B. Težak, "A simulation of heat transfer during billet transport," *Applied Thermal Engineering*, vol. 22, no. 7, pp. 873–883, 2002.
- [4] B. Zhang, Z. G. Chen, and L. Y. Xu, "The modeling and control of a reheating furnace," in *Proceedings of the American Control Conference*, 2002.
- [5] B. Zhang, J. C. Wang, and J. M. Zhang, "Dynamic model of reheating furnace based on fuzzy system and genetic algorithm," *Control Theory & Application*, vol. 20, no. 2, pp. 293–296, 1998 (Chinese).
- [6] H. J. Wick, "Estimation of ingot temperature in a soaking pit using an extended Kalman filter," in *Proceedings of the 8th Triennial World Congress of the International Federation of Automatic Control*, 1981.
- [7] D. Xiao, Y. H. Yang, and Z. Z. Mao, "A model for billet temperature of prediction of heating-furnace based on improved PCR method," *Information and Control*, vol. 34, no. 3, pp. 340–343, 2005 (Chinese).
- [8] Y.-W. Chen and T.-Y. Chai, "Preprocessing of operation data in heating furnace," *Control Theory and Applications*, vol. 29, no. 1, pp. 114–118, 2012 (Chinese).
- [9] G. M. Cui and G. B. Ding, "Research on the optimal control of tube billet temperature for rotary reheating furnace," in *Advanced Electrical and Electronics Engineering*, vol. 87 of *Lecture Notes in Electrical Engineering*, pp. 471–477, Springer, Berlin, Germany, 2011.
- [10] H. Iwamoto, O. Sugiyama, R. Nakanishi, and T. Okuyama, "Automatic control system of billet reheating rotary hearth furnace," in *Proceedings of the International Conference on Industrial Electronics, Control, Instrumentation*, 1992.
- [11] F. He, A. Xu, H. Wang, D. He, and N. Tian, "End temperature prediction of molten steel in LF based on CBR," *Steel Research International*, vol. 83, no. 11, pp. 1079–1086, 2012.
- [12] W. Lv, Z. Mao, and P. Yuan, "Ladle furnace steel temperature prediction model based on partial linear regularization networks with sparse representation," *Steel Research International*, vol. 83, no. 3, pp. 288–296, 2012.
- [13] S. Wold, N. Kettaneh-Wold, and B. Skagerberg, "Nonlinear PLS modeling," *Chemometrics and Intelligent Laboratory Systems*, vol. 7, no. 1-2, pp. 53–65, 1989.
- [14] S. J. Qin, "Recursive PLS algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, 1998.
- [15] B. Hu, Z. Zhao, and J. Liang, "Multi-loop nonlinear internal model controller design under nonlinear dynamic PLS framework using ARX-neural network model," *Journal of Process Control*, vol. 22, no. 1, pp. 207–217, 2012.
- [16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.
- [17] Y. Yu, T.-M. Choi, and C.-L. Hui, "An intelligent quick prediction algorithm with applications in industrial control and loading problems," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 276–287, 2012.
- [18] J. Zhai, H. Xu, and Y. Li, "Fusion of extreme learning machine with fuzzy integral," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 21, supplement 2, pp. 23–34, 2013.
- [19] J.-H. Zhai, H.-Y. Xu, and X.-Z. Wang, "Dynamic ensemble extreme learning machine based on sample entropy," *Soft Computing*, vol. 16, no. 9, pp. 1493–1502, 2012.
- [20] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, June 2014.
- [21] Y. Jin, J. W. Cao, Q. Q. Ruan, and X. Q. Wang, "Cross-modality 2D-3D face recognition via multiview smooth discriminant analysis based on ELM," *Journal of Electrical and Computer Engineering*, vol. 2014, Article ID 584241, 9 pages, 2014.
- [22] J. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [23] Y. Yang, Y. Wang, and X. Yuan, "Bidirectional extreme learning machine for regression problem and its learning effectiveness," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1498–1505, 2012.
- [24] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [25] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.
- [26] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, 2014.
- [27] H.-X. Tian and Z.-Z. Mao, "An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 73–80, 2010.
- [28] G. Feng, Z. Qian, and N. Dai, "Reversible watermarking via extreme learning machine prediction," *Neurocomputing*, vol. 82, no. 4, pp. 62–68, 2012.
- [29] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feed forward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [30] J. Zhao, Z. Wang, and D. S. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, pp. 79–89, 2012.
- [31] S. J. Xie, J. Yang, H. Gong, S. Yoon, and D. S. Park, "Intelligent fingerprint quality analysis using online sequential extreme learning machine," *Soft Computing*, vol. 16, no. 9, pp. 1555–1568, 2012.
- [32] M. Khalid, S. Omatu, and R. Yusof, "MIMO furnace control with neural networks," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 4, pp. 238–245, 1993.
- [33] C.-H. Lu, C.-C. Tsai, C.-M. Liu, and Y.-H. Charnng, "Neural-network-based predictive controller design: an application to temperature control of a plastic injection molding process," *Asian Journal of Control*, vol. 12, no. 6, pp. 680–691, 2010.

## Research Article

# Efficient ELM-Based Two Stages Query Processing Optimization for Big Data

Linlin Ding,<sup>1</sup> Yu Liu,<sup>1</sup> Baoyan Song,<sup>1</sup> and Junchang Xin<sup>2</sup>

<sup>1</sup>School of Information, Liaoning University, Shenyang, Liaoning 110036, China

<sup>2</sup>College of Information Science & Engineering, Northeastern University, Shenyang, Liaoning 110819, China

Correspondence should be addressed to Baoyan Song; [bysong@lnu.edu.cn](mailto:bysong@lnu.edu.cn)

Received 22 August 2014; Accepted 27 November 2014

Academic Editor: Yi Jin

Copyright © 2015 Linlin Ding et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

MapReduce and its variants have emerged as viable competitors for big data analysis with a commodity cluster of machines. As an extension of MapReduce, ComMapReduce realizes the lightweight communication mechanisms to enhance the performance of query processing applications for big data. However, different communication strategies of ComMapReduce can substantially affect the executions of query processing applications. Although there is already the research work that can identify the communication strategies of ComMapReduce according to the characteristics of the query processing applications, some drawbacks still exist, such as relative simple model, too much user participation, and relative simple query processing execution. Therefore, an efficient ELM-based two stages query processing optimization model is proposed in this paper, named ELM to ELM (*E2E*) model. Then, we develop an efficient sample training strategy to train our *E2E* model. Furthermore, two query processing executions based on the *E2E* model, respectively, Just-in-Time execution and Queue execution, are presented. Finally, extensive experiments are conducted to verify the effectiveness and efficiency of the *E2E* model.

## 1. Introduction

Nowadays, MapReduce [1] has become a widespread programming framework for big data analysis. Hadoop (<http://hadoop.apache.org/>) is a popular open-source implementation of MapReduce that many academies and industrial organizations adopt it in research and production deployments. The success of MapReduce stems from hiding the details of parallelization, fault-tolerance, and load balancing in a simple programming framework. MapReduce and its variants are used to process the big data applications, such as Web indexing, data mining, machine learning, financial analysis, scientific simulation, and bioinformatics research [2–10].

As one of the successful extensions of MapReduce, ComMapReduce [3, 4] adds simple lightweight communication mechanisms to generate the certain *shared information* and implements the query processing applications for big data. In ComMapReduce framework, three basic and two optimization communication strategies are presented to solve the problem of how to communicate and obtain the *shared*

*information* of different applications. After further analyzing the ComMapReduce execution course and the abundant experiments, we find out that different communication strategies of ComMapReduce can substantially impact the performance of query processing applications.

To identify the communication strategy, the existing research work [11] proposes a query optimization model, named *ELM.CMR*, based on ELM [12] which has the classification performance at an excellent standard. According to the characteristics of query processing programs, *ELM.CMR* can identify the communication strategy of ComMapReduce. However, *ELM.CMR* still has the following drawbacks. First, regardless of the MapReduce or ComMapReduce framework, a program only consists of black\_box Map and Reduce functions, without knowing the distributed details about the framework. But the configuration parameters of the framework can fully influence the performance of query processing. Finding the most suitable configuration parameters setting itself is difficult. The burden falls on the user who submits the MapReduce or ComMapReduce job to specify

settings for all configuration parameters, which highlights the challenges the user faces. How to identify the proper configuration parameters according to the user program is a difficult problem. Second, the implementation algorithm of multiple queries in *ELM\_CMR* only processes the queries based on the classification results and predicted execution time, but there is no consideration of the characteristics among different queries. Different queries can share computation and data so as to enhance the performance further. Third, *ELM\_CMR* only adopts simple training method to gain the classification model. If more efficient training method is adopted to obtain the training data, the accuracy of the classification model can be improved.

Therefore, in this paper, for resolving the above problems and drawbacks, we propose an efficient ELM-based two stages query processing optimization model, named ELM to ELM (*E2E*) model. According to the characteristics of the users programs, the first stage can gain the *feature parameters* using ELM algorithm. After that, according to the results of the first stage, the second stage can identify the final classification result by ELM algorithm too. Furthermore, an efficient sample training strategy is presented to train the *E2E* model. We also develop two query processing executions based on the *E2E* model, Just-in-Time execution and Queue execution. The contributions of this paper can be summarized as follows.

- (i) We propose an efficient ELM-based two stages query processing optimization model, named *E2E* model, which can realize the most optimal executions of query processing applications in MapReduce or ComMapReduce framework.
- (ii) We develop a sample training strategy to train our *E2E* model and two query executions based on *E2E* model, Just-in-Time execution, and Queue execution.
- (iii) The experimental studies using synthetic data show the effectiveness and efficiency of the *E2E* model.

The remainder of this paper is organized as follows. Section 2 briefly introduces the background, containing the ELM and *ELM\_CMR*. Our *E2E* model is proposed in Section 3. The two executions for query processing applications based on *E2E* are presented in Section 4. The experimental results to show the performance of *E2E* model are reported in Section 5. Finally, we conclude this paper in Section 6.

## 2. Background

In this section, we describe the background of our work, which includes a brief overview of the traditional ELM and the detailed descriptions of the *ELM\_CMR*.

**2.1. Review of ELM.** Nowadays, extreme learning machine (ELM) [12] and its variants [13–28] have the characteristics of excellent generalization performance, rapid training speed, and little human intervene, which have attracted extensive attention from more and more researchers. ELM is originally designed for the single hidden-layer feedforward neural

networks (SLFNs [29]) and is then extended to the “generalized” SLFNs. ELM algorithm first randomly allocates the input weights and hidden layer biases and then analytically computes the output weights of SLFNs. Contrary to the other conventional learning algorithms, ELM reaches the optimal generalization performance at a very fast learning speed. ELM is less sensitive to the user defined parameters, so it can be deployed fast and convenient. That is the reason we choose ELM as our basic method to construct the query processing optimization model.

For  $N$  arbitrary distinct samples  $(\mathbf{x}_j, \mathbf{t}_j)$ , where  $\mathbf{x}_j = [x_{j1}, x_{j2}, \dots, x_{jm}]^T \in \mathbb{R}^n$  and  $\mathbf{t}_j = [t_{j1}, t_{j2}, \dots, t_{jm}]^T \in \mathbb{R}^m$ , standard SLFNs with hidden nodes  $L$  and activation function  $g(x)$  are mathematically modeled as

$$\sum_{i=1}^L \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^L \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j \quad (j = 1, 2, \dots, N), \quad (1)$$

where  $L$  is the number of hidden layer nodes,  $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{im}]^T$  is the weight vector between the  $i$ th hidden node and the input nodes,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the weight vector connecting the  $i$ th hidden node and the output nodes,  $b_i$  is the threshold of the  $i$ th hidden node, and  $\mathbf{o}_j = [o_{j1}, o_{j2}, \dots, o_{jm}]^T$  is the  $j$ th output vector of the SLFNs.

The standard SLFNs can approximate these  $N$  samples with zero error. The error of ELM is  $\sum_{j=1}^L \|\mathbf{o}_j - \mathbf{t}_j\| = 0$  and there exist  $\beta_i$ ,  $\mathbf{w}_i$ , and  $b_i$  such that

$$\sum_{i=1}^L \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j \quad (j = 1, 2, \dots, N). \quad (2)$$

Equation (2) can be expressed compactly as follows:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \quad (3)$$

where

$$\begin{aligned} \mathbf{H}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_L, b_1, b_2, \dots, b_L, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L) \\ &= \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_N) \end{bmatrix} \\ &= \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & g(\mathbf{w}_2 \cdot \mathbf{x}_1 + b_2) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ g(\mathbf{w}_1 \cdot \mathbf{x}_2 + b_1) & g(\mathbf{w}_2 \cdot \mathbf{x}_2 + b_2) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_2 + b_L) \\ \vdots & \vdots & \vdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & g(\mathbf{w}_2 \cdot \mathbf{x}_N + b_2) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L}, \end{aligned} \quad (4)$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \quad \mathbf{T} = \begin{bmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}. \quad (5)$$

$\mathbf{H}$  is set as the hidden layer output matrix of the neural network. The  $i$ th column of  $\mathbf{H}$  is called the  $i$ th hidden node

output with respect to inputs  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ . The smallest norm least-squares solution of the above multiple regression system is shown as follows:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{T}, \quad (6)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ . Then the output function of ELM can be modeled as follows:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \boldsymbol{\beta} = \mathbf{h}(\mathbf{x}) \mathbf{H}^\dagger \mathbf{T}. \quad (7)$$

The computational course for ELM is shown in Algorithm 1. Only after properly setting the related parameters, ELM can start the training process. The first step is to generate  $L$  pairs of hidden node parameters  $(\mathbf{w}_i, b_i)$  (Lines 1–3). The second step actually calculates the hidden layer output matrix  $\mathbf{H}$  by using (4) (Line 4). The third step mainly computes the corresponding output weight vector  $\boldsymbol{\beta}$  (Line 5). After completing the above training process, the output of the new dataset can be predicted by ELM according to (7).

**2.2. ELM-CMR Model.** ComMapReduce [3, 4] is an improved MapReduce framework with lightweight communication mechanisms. A new node, named the Coordinator node, is added to store and generate the certain *shared information* of different applications. In ComMapReduce, three basic communication strategies, LCS, ECS, and HCS, and two optimization communication strategies, PreOS and PostOS, are proposed to identify how to receive and generate the *shared information*. In short, without affecting the existing characteristics of the original MapReduce framework, ComMapReduce is a successful parallel programming framework with global *shared information* to filter the unpromising data of query processing programs.

The existing *ELM-CMR* [11] is an efficient query processing optimization model based on ELM. It can identify the communication strategies of query processing applications in ComMapReduce according to the features of queries. Figure 1 shows the architecture of *ELM-CMR* model. The four components of *ELM-CMR* are, respectively, the *Feature Selector*, the *ELM Classifier*, the *Query Optimizer*, and the *Execution Fabric*.

The *Feature Selector* mainly examines the training query processing programs and selects the configuration parameters that can wholly affect the query performance by the job profiles. Naturally, the parameters of program  $p$  can be divided into three types, parameters that predominantly affect Map task execution; parameters that predominantly affect Reduce task execution; and the cluster parameters. Then, in each cluster, we adopt the minimum-redundancy-maximum-relevance (mRMR) [30] feature selection to find the optimal parameters sharply affecting the performance. And then, we generate the globally optimal configuration parameter settings by combining the results of each subspace. Therefore, the near-optimal configuration parameter setting can be generated.

After selecting the features of training data, the *Feature Selector* sends the extracted training data to the *ELM Classifier*. It uses the training data to construct the ELM model by the traditional ELM algorithm. After that, when there are one or multiple queries to be processed, the *ELM Classifier* can rapidly obtain the classification results of the queries and then sends them to the *Query Optimizer*.

The *Query Optimizer* applies the classification results of the *ELM Classifier* and combines the implementation patterns to choose an optimized *execution order*. After gaining the *execution order*, the query is sent to the *Execution Fabric*.

The *Execution Fabric* implements the program in ComMapReduce framework. When there is one query to be processed, the *Execution Fabric* implements the query according to the classification result of the *Query Optimizer* in *ELM-CMR*. When there are multiple queries to be processed, the multiple queries can be classified by *ELM Classifier* and gain the best communication strategy of each program. Then, a Task Scheduler Simulator is used to simulate the execution time of queries. According to the execution time and the classification results of the queries, the *Query Optimizer* designs an *execution order* following the common principle of Shortest Job First (*SJF*) to implement multiple queries.

### 3. E2E Model

In this section, the overview of our *E2E* model is introduced first in Section 3.1. Then, training the *E2E* model is shown in Section 3.2. Finally, predicting the *E2E* model is proposed in Section 3.3.

**3.1. Overview of E2E Model.** Our *E2E* model can identify the optimal communication strategies of query processing programs in MapReduce or ComMapReduce, which contains three main phases, respectively, the *training phase*, the *prediction phase*, and the *execution phase*. Figure 2 shows the whole workflow of query processing in the *E2E* model. The main workflow is as follows.

First, the *training phase* is responsible for extracting the training query processing programs that have a large effect on the *E2E* model. In the *training phase*, we use sample-based training strategies to run our workload, in isolation, pairwise, and at several higher multiprogramming levels. After gaining the training samples, they can be used to generate the *E2E* model in the *prediction phase*. The details of training phase will be introduced in Section 3.2.

Second, in the *prediction phase*, by using the training samples from the *training phase*, the two stages *E2E* model can be generated based on the traditional ELM algorithm. According to the user's programs, the first stage can obtain the most optimal *feature parameters* of the programs by ELM. And then, using the *feature parameters* of the first stage, the second stage can identify the classification results of the query processing programs by ELM. The details of the *prediction phase* will be shown in Section 3.3.

Third, after gaining the *E2E* model, for the query processing programs submitted to MapReduce or ComMapReduce,

```

(1) for  $i = 1$  to  $L$  do
(2)   Randomly generate the hidden node parameters  $(w_i, b_i)$ ;
(3) end for
(4) Calculate the hidden layer output matrix  $\mathbf{H}$ ;
(5) Calculate the output weight vector  $\beta = \mathbf{H}^+ \mathbf{T}$ ;
    
```

ALGORITHM 1: ELM.

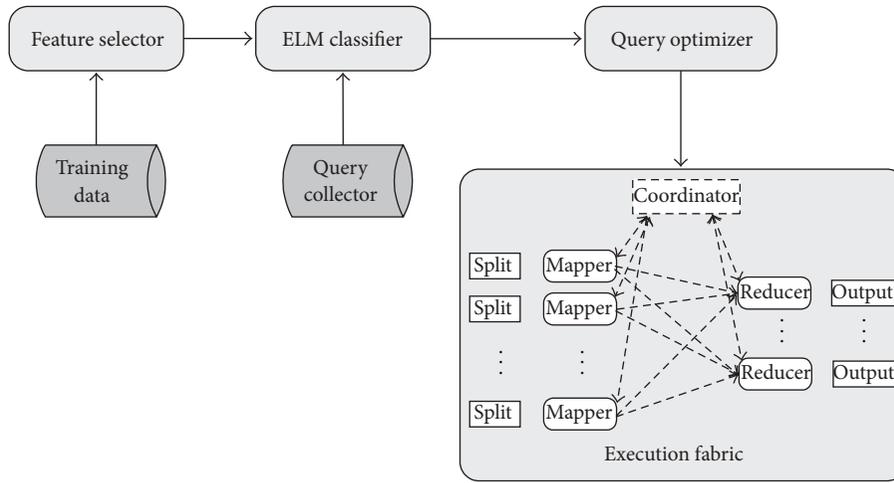


FIGURE 1: Architecture of *ELM\_CMR* model.

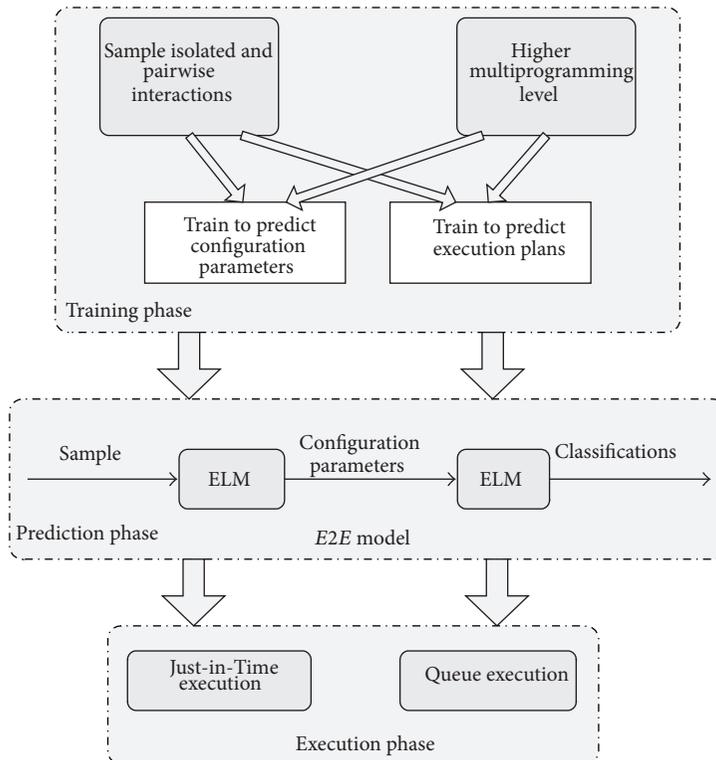


FIGURE 2: Workflow of *E2E* model.

we can predict their optimal communication strategies in the *execution phase*. We propose two query executions based on *E2E* model, the Just-in-Time execution which tackles one query processing program, and the Queue execution which tackles multiple concurrent queries. The details of the execution phase will be illustrated in Section 4.

**3.2. Training E2E Model.** To obtain the prediction model more accurately, we need to train the *E2E* model. Different from the simple training course of *ELM\_CMR* model, the *training phase* of our *E2E* model consists of running the queries in isolation, pairwise as well as at several higher multiprogramming levels. In addition, in order to gain the measurements that capture the interactions, we omit the first few samples in our experiments. This approach makes us ignore the overhead of the initial query setup and caching the initial supporting structures.

The *E2E* model realizes the *training phase* by sampling approach, containing isolation, pairwise, and higher degree of concurrency, which allows us to approximate the running of queries in MapReduce or ComMapReduce. The course of the *training phase* is displayed in Figure 2. First, we sample the workload in isolation to gain how each query behaves, which can be seen as the baseline of training. Second, according to the query types in our experiments, we build a matrix of interaction by running all unique pairwise combinations. Pairwise sample can help us to simply estimate the degree of concurrency. Here, Latin hypercube sampling approach (LHS) can uniformly distribute our samples throughout our prediction space, which can be realized by creating a hypercube with the same dimension as the multiprogramming level. We adopt LHS to sample at pairwise or several higher multiprogramming levels. We then select the samples that every value on every plane gets inter selected exactly one. A simple example of two-dimensional LHS is shown in Table 1. In Table 1, Query-A, Query-B, Query-C, and Query-D stand for four queries to be processed by pairwise combinations. After combining using LHS, the four queries can obtain the optimal combinations.

Furthermore, the above strategy shows how to run the samples, in isolation, pairwise, and at several higher multiprogramming levels using LHS. As we know, the number of queries to be processed is large, and in order to enhance the training accuracy, we need to choose the queries that is the representative queries in each query type and then construct the samples to be trained. The corresponding cluster algorithms based on DBSCAN [31] can be also adopted to gain the most representative query in each query type. In practice, although the training time of *E2E* may be long, it is worth mentioning that *training phase* allows our model to be extremely lightweight once it reaches the *prediction phase* and enhance the accuracy.

**3.3. Predicting E2E Model.** Then, we introduce the details of the *prediction phase*. A MapReduce job  $j$  can be expressed by a MapReduce program  $p$  running on input data  $d$  and cluster  $r$ , which can be expressed as  $j = \langle p, d, r, c \rangle$  in short. We call the  $d$ ,  $r$ , and  $c$  the *feature parameters* of  $p$ . The users only

TABLE 1: An example of LHS.

Query	Query-A	Query-B	Query-C	Query-D
Query-A		▲		
Query-B			▲	
Query-C	▲			
Query-D				▲

TABLE 2: Feature parameters in the experiments.

Property name	Type	Default value
io.sort.mb	int	100
io.sort.factor	int	10
min.num.spills.for.combine	int	3
mapred.compress.map.output	boolean	false
mapred.reduce.parallel.copies	int	5
mapred.reduce.copy.backoff	int	300
dfs.heartbeat.interval	int	3
dfs.block.size (M)	int	64
mapred.map.task	int	4
mapred.reduce.task	int	4
mapred.tasktracker.map.task.maximum	int	4
mapred.tasktracker.reduce.task.maximum	int	4
Data size (G)	int	10
Data distribution	boolean	uniform
Number of slave nodes	int	8

submit their jobs to MapReduce or ComMapReduce without knowing the internal configuration details of the system.

However, a number of choices can be made in order to fully specify how the job should be executed. These choices, represented by  $c$  in  $\langle q, d, r, c \rangle$ , stand for a high dimensional space of configuration parameter settings, such as the number of Map and Reduce task, the block size, and the amount of memory. The performance changes a lot in different configuration parameters. For any parameter, its value is not specified explicitly during job submission, the default values either shipped with the system or specified by the system administrator. However, the normal users do not understand the running details of MapReduce. That is to say, finding good configuration settings for MapReduce job is time consuming and requires extensive knowledge of system internals. Because the users have little information of the parallelization details of MapReduce, it is necessary to gain the suitable configuration parameters. The first stage of our *E2E* model is to generate the model for identifying the suitable configuration parameters of query processing programs by ELM. The main goal of the first stage is to construct a *black\_box feature parameters* of queries, containing  $\langle d, r, c \rangle$ . The configuration parameters wholly affecting the performance adopted by *E2E* are the same as our *ELM\_CMR* approach as shown in Table 2. The corresponding information of the job can be obtained from sampling a few tasks of the job. After the first stage, the *feature parameters* setting can be obtained by ELM algorithm.

- (1) Generate the *feature parameters* of  $j$  by the first stage of  $E2E$  model;
- (2) Generate the execution plan of  $j$  by the second stage of  $E2E$  model;
- (3) Execute  $j$  with its communication strategy;

ALGORITHM 2: Just-in-Time execution.

After gaining the configuration parameters, the second stage is to use them to predict the communication strategy of ComMapReduce or MapReduce by ELM algorithm too and then generates the classification results. This stage is similar to the prediction of *ELM\_CMR*, so we do not illustrate in detail in this section.

#### 4. Executions of $E2E$ Model

After generating the  $E2E$  model, the pending queries can be implemented under our  $E2E$  model. In this section, two scenarios are proposed for executing query processing programs using the  $E2E$  model. In the first scenario, one new query is being submitted for immediate execution, named as Just-in-Time execution. In the second scenario, a Queue-based multiple concurrent queries execution is proposed, where an ordered list of queries to run is gained.

**4.1. Just-in-Time Execution.** When there is one new query being submitted, the  $E2E$  model generates its classification result as soon as possible. According to the characteristics of the coming query, the first stage of  $E2E$  model can generate the *feature parameters* of this query based on ELM. After that, the user can make a decision that whether adopting ComMapReduce or which communication strategy can be adopted by the second stage of  $E2E$  model, and then implements the query processing application.

The course of Just-in-Time execution is shown in Algorithm 2. First, the most optimal *feature parameters* of query processing job  $j$  are extracted in the first stage of  $E2E$  model (Line 1). Second, after obtaining the *feature parameters* of job  $j$ , the  $E2E$  generates the classification result of  $j$  (Line 2). Third, according to the classification of  $j$ , the  $E2E$  ensures how to implement the query and sends it to MapReduce or ComMapReduce framework. The framework uses the optimization result to execute the query program (Line 3).

For example, for a submitted skyline query, the first stage of  $E2E$  can obtain the most optimal *feature parameters* of this skyline query. After abstracting its *feature parameters*, the  $E2E$  can generate its classification and then identifies the communication strategy of this skyline query, such as PostOS. After that, the skyline query will be implemented in ComMapReduce with PostOS.

**4.2. Queue Execution.** When there are multiple concurrent queries to be processed, we design an efficient *execution order* of queries based on  $E2E$  model. Under the *execution order*, the performance of multiple concurrent queries can reach a nice status. Of course, we can execute these queries one by one

with their classification results of  $E2E$ , but a lot of calculations and execution time would be wasted. Therefore, the *execution order* is important to enhance the performance of multiple concurrent queries.

As we know, different queries usually contain the overlap parts, such as the same query type and the same processing data. Therefore, when processing multiple concurrent queries, if we can identify the correlations of query type and processing data of the multiple queries, then the sharing computation and data can be used to prevent the repeated processing and scanning.

The main processing course of Queue execution is shown in Figure 3. There are three main components in the Queue execution, respectively, the *Analyzer*, the *Optimizer*, and the *Executor*.

First, for the multiple concurrent queries to be executed, after obtaining the classification results by  $E2E$  model, the *Analyzer* analyzes the queries and divides the queries into some subsets by considering the share of computation. This dividing approach can divide the queries with the same query type into the same subset, then the queries in a subset can be computed together.

Second, in each subset, the *Optimizer* can design the local execution order according to the characteristics of the sharing computation and data. It can analyze the data being processed of the multiple concurrent queries in the same subset, and then the queries with sharing data can share the computation results of the sharing data. The *Optimizer* also can cache the corresponding intermediate results into memory or buffer by the characteristics of the queries so as to reduce the repeated implementations of the same operations and I/O cost. Then, similar to our former *ELM\_CMR*, the *Optimizer* also uses a Task Scheduler Simulator to simulate the scheduling and executions of Map and Reduce tasks of each query. The execution time of a job  $j$  can be estimated by the Task Scheduler Simulator. So, according to the common principle of Shortest Job First (*SJF*) too, the *execution order* of each subset is gained.

Finally, after obtaining the *execution order* of each subset, the *Executor* also uses the *SJF* principle of the subsets by simple merging their execution time of each subset, so the global execution order ( $O_s$ ) of the multiple queries is the ascending order of the simulated execution time of each subset. Then, the multiple queries can be implemented under the global *execution order*.

Algorithm 3 illustrates the whole execution course of the multiple concurrent queries. First, we can obtain the classification of each query by the  $E2E$  model (Lines 1–3). Then, the *Analyzer* divides the queries sharing computation into some subsets (Line 4). Then, in each subset, the *Optimizer*

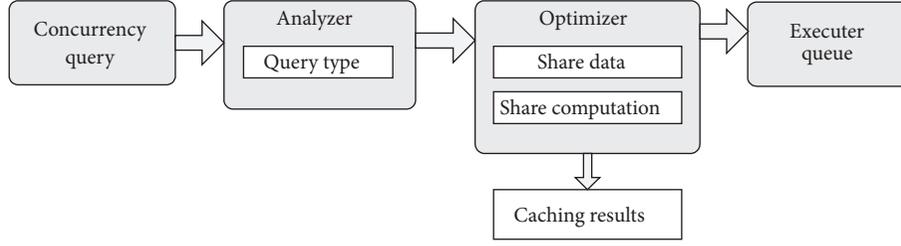


FIGURE 3: Workflow of Queue execution.

- (1) **for** each query  $q_i$  **do**
- (2) Generate the classification of  $q_i$  by *E2E*;
- (3) **end for**
- (4) Divide the queries into some subsets by query type;
- (5) **for** each subset **do**
- (6) Analyze and divide the sharing data;
- (7) Compute and query of the sharing data;
- (8) Calculate the simulate execution time;
- (9) Generate the local *execution order* of each subset by *SJF*;
- (10) **end for**
- (11) Generate the global *execution order* by *SJF*;

ALGORITHM 3: Multiple queries.

can analyze and divide the sharing data, and then computes the sharing data. After gaining the simulated time of each query in each subset, the local *execution order* is obtained by considering the sharing and features of the queries in each subset by *SJF* (Lines 5–10). Finally, the global *execution order* of the queries is generated by *SJF* too (Line 11).

Figure 4 shows an example of the implementation of the multiple concurrent queries. Suppose that there are eight queries to be processed. First, these queries can obtain their classification results by *E2E*. After that, the *Analyzer* divides these queries into four subsets by the sharing of computation, the same query type, respectively, top- $k$ ,  $k$ NN, skyline, and join.

In each subset, the *Optimizer* can make a local *execution order* by the features of the queries. For example, the subset of skyline query contains three skyline queries,  $q_2$ ,  $q_5$ , and  $q_8$ , with their processing data sets  $s_2 = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ ,  $s_5 = \{p_1, p_2, p_3, p_7, p_8, p_9\}$ , and  $s_8 = \{p_4, p_5, p_6, p_7, p_8, p_9\}$ . For these skyline queries, the *Optimizer* divides the sharing data and gains the following sharing subdatasets,  $s_a = \{p_1, p_2, p_3\}$ ,  $s_b = \{p_4, p_5, p_6\}$ , and  $s_c = \{p_7, p_8, p_9\}$ . Then, the *Optimizer* can transfer the above skyline queries into some simple skyline queries,  $s_2 = s_a \cup s_b$ ,  $s_5 = s_a \cup s_c$ , and  $s_8 = s_b \cup s_c$ . So, the *Optimizer* can make a solution that the skyline queries first compute the results of  $s_a$ ,  $s_b$ , and  $s_c$ , and then gain a local *execution order* by *SJF*,  $q_5$ ,  $q_8$ , and  $q_2$ . The similar processing course is fit for the top- $k$  and  $k$ NN queries. For our join query of small-big tables, the *Optimizer* can cache the corresponding small table into memory of Map tasks so as to reduce the repeated scan of the small table. Finally, the

global *execution order*  $O_s, q_7, q_1, q_5, q_8, q_2, q_3, q_4$ , and  $q_6$  can be gained by simple merge of *SJF* of the four subsets.

## 5. Performance Evaluation

In this section, the performance of our *E2E* model is evaluated in detail with various experimental settings. We first describe the setup used in our experiments in Section 5.1. Then we present and discuss the experimental results in Section 5.2.

**5.1. Experimental Setup.** The experimental setup is the same as *ELM\_CMR* model as follows. The experimental setup is a Hadoop cluster running on 9 nodes in a high speed Gigabit network, with one node as the Master node and the Coordinator node and the others as the Slave nodes. Each node has an Intel Quad Core 2.66 GHz CPU, 4 GB memory, and CentOS Linux 5.6. We use Hadoop 0.20.2 and compile the source codes under JDK 1.6. The *ELM* algorithm is implemented in MATLABR2009a.

*ELM\_CMR* is an efficient query processing optimization model based on *ELM*. It can identify the communication strategies of query processing applications in ComMapReduce according to the features of queries. So far, except the *ELM\_CMR* model, there is no any other method to study how to choose the optimal strategies of ComMapReduce for query processing applications. So, we evaluate the performance of *E2E* model in different implementations for Just-in-Time execution and Queue execution by comparing with *ELM\_CMR*.

In order to fully evaluate the performance of *E2E* model, four typical query processing applications are adopted to

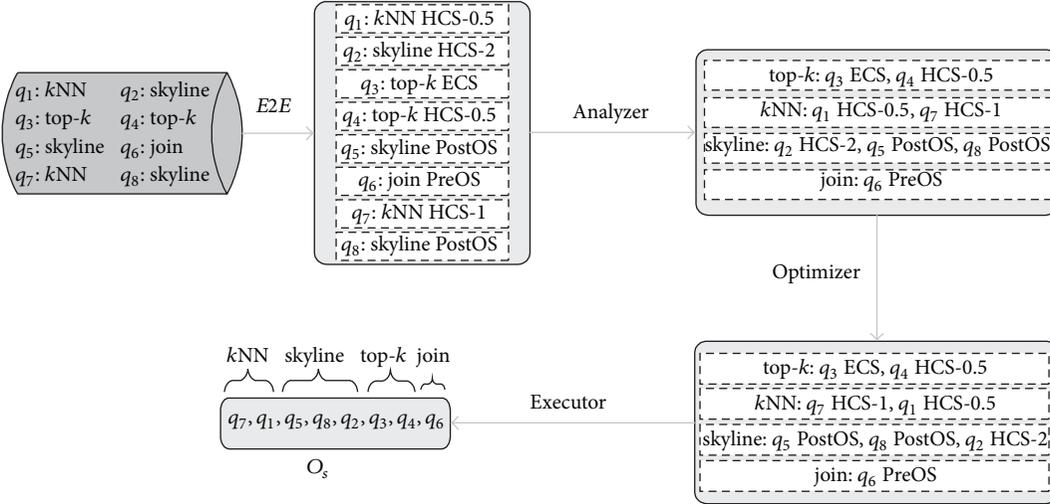


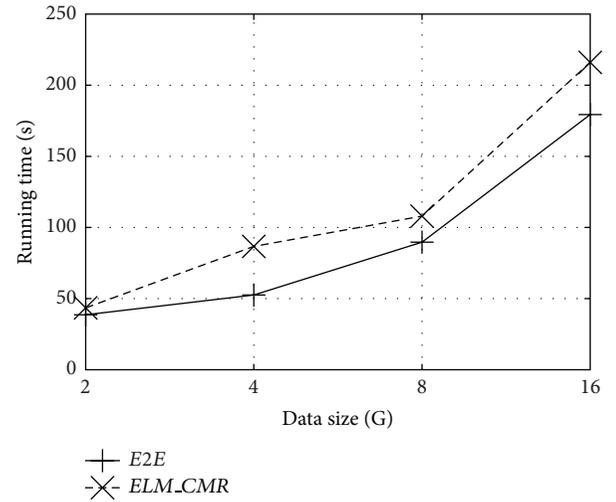
FIGURE 4: Implementation of multiple queries.

evaluate the implementations of Just-in-Time execution and Queue execution, respectively, top- $k$ , kNN, skyline, and join. In Just-in-Time execution evaluating, we, respectively, test the performance of top- $k$ , kNN, skyline, and join under different data sizes in *E2E* and *ELM\_CMR*. In Queue execution evaluating, we design two kinds of experiments. One is that the multiple queries are the same kind of query under different data sizes, taking skyline query as an example. The other one is that the multiple queries are the different kinds of queries under different data sizes, taking top- $k$  (2 G), top- $k$  (8 G), kNN (2 G), and kNN (8 G) as examples.

We use the synthetic data to test the performance. The synthetic datasets include different data sizes and different distributions, such as uniform distribution in top- $k$  and kNN, anticorrelated distribution in skyline, and small-big tables in join. They can reflect the performance under different situations. The classification results of *E2E* model contain 7 types, respectively, ECS, HCS-0.5, HCS-1, HCS-2, PreOS, PostOS, and MapReduce (MR). HCS-0.5 means the preassigned time interval of HCS is 0.5 s. Table 2 summarizes the parameters used in the experiments including the default values.

**5.2. Experimental Results.** First, we give the classification results of *E2E* model and *ELM\_CMR* model by processing the above four query processing applications in different data size shown in Table 3. We can see that for the same query type and data size, the classification results of the queries are different, so the performance of the two models is different too. In the following, we evaluate the performance of the queries according to the classification results of Table 3.

Second, the performance of Just-in-Time execution is shown in the following figures. Figure 5 shows the performance of top- $k$  queries ( $k = 1000$ ), with the data size is 2 G, 4 G, 8 G, and 16 G in uniform distribution. We can see that the performance of top- $k$  queries in the classification results of *E2E* model is better than *ELM\_CMR*. The reason is that *E2E* model can effectively evaluate the optimal configuration parameters of the queries than *ELM\_CMR* model. When  $k$  is

FIGURE 5: Performance of top- $k$  query.

much smaller than the original data, the global *shared information* of ComMapReduce can reach the most optimal one quickly, so the Mappers can retrieve the *shared information* in the initial phase to filter the unpromising data.

Figure 6 shows the performance of kNN queries with different data size of uniform distribution. The performance of *E2E* is also optimal to *ELM\_CMR* model, where the reason is that *E2E* model can gain the suitable classification results.

Figure 7 shows the performance of skyline queries in anticorrelated distribution with different data size. We can see that the performance of different execution plans is not obviously different, but PostOS is a little better. The reason is that the original data are skewed to the final results in anticorrelated distribution. The percentage of filtering is low, so the performance difference is not obvious. In this situation, although *E2E* and *ELM\_CMR* can obtain the classification, it can also choose the other communication strategies.

TABLE 3: Classifications of E2E and ELM\_CMR.

Query	top-k		kNN				skyline				join					
	2	4	8	16	2	4	8	16	0.2	0.4	0.8	1.6	5	10	15	20
Data (G)	HCS-0.5	HCS-0.5	PreOS	PreOS	ECS	HCS-0.5	HCS-0.5	PreOS	HCS-0.5	HCS-0.5	PostOS	PostOS	ECS	ECS	ECS	PreOS
E2E	HCS-0.5	HCS-0.5	HCS-1	HCS-0.5	HCS-0.5	HCS-1	HCS-2	HCS-2	ECS	HCS-1	HCS-2	HCS-2	MR	HCS-1	HCS-2	HCS-2
ELM_CMR	ECS	ECS	HCS-1	HCS-0.5	HCS-0.5	HCS-1	HCS-2	HCS-2	ECS	HCS-1	HCS-2	HCS-2	MR	HCS-1	HCS-2	HCS-2

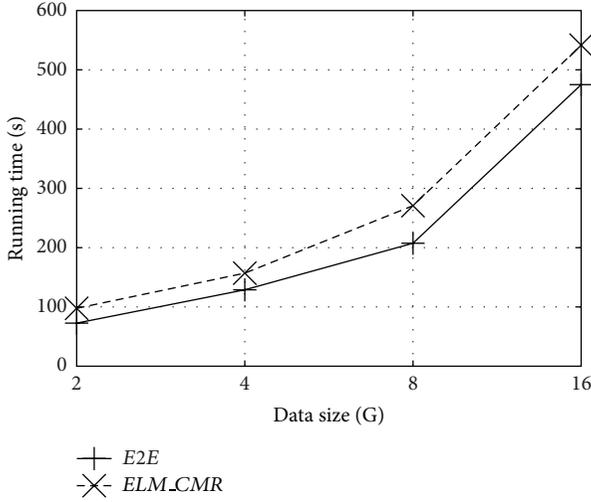
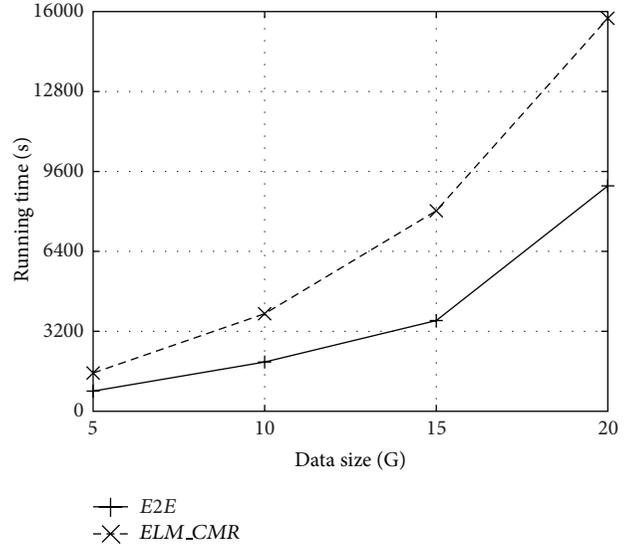
FIGURE 6: Performance of  $k$ NN query.

FIGURE 8: Performance of join query.

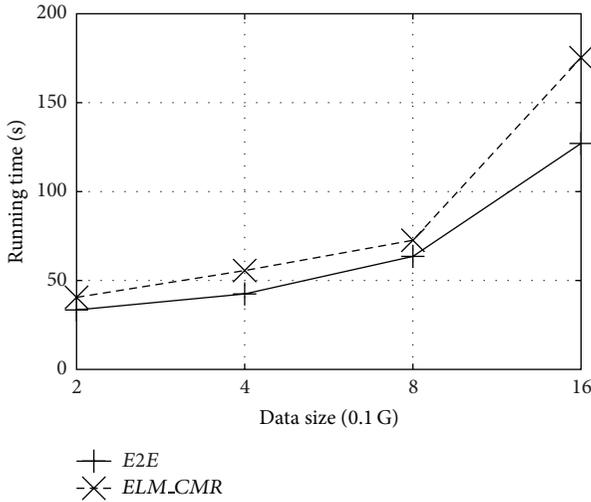


FIGURE 7: Performance of skyline query.

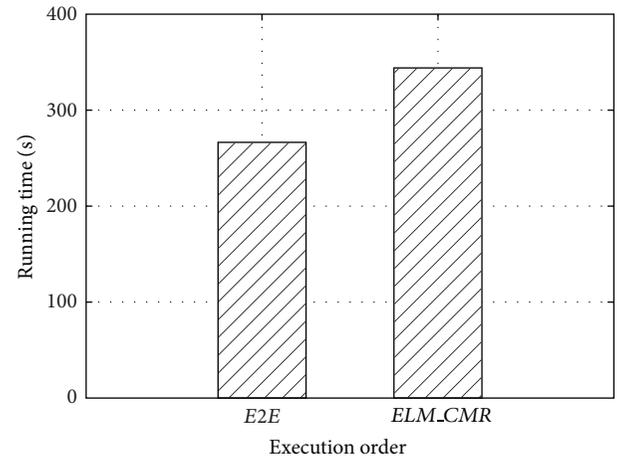


FIGURE 9: The same query type.

Figure 8 shows the performance of join queries in different data size of small-big tables, with the same data size of the small table 2 G, and the different data sizes of big table are shown in Table 3. The performance of  $E2E$  is much better than  $ELM\_CMR$ . In ComMapReduce, the join attributes of the small table can be set as the *shared information* to filter the unpromising intermediate results.

Third, we evaluate the performance of the same kind of multiple queries implemented in different *execution orders*. Figure 9 illustrates the performance of two *execution orders* of four skyline queries in different data sizes, 0.2 G, 0.4 G, 0.8 G, and 1.6 G in anticorrelated distribution. The running time of our optimized *execution order* is shorter than the running time of the original order. Our  $E2E$  can identify the proper classifications and *execution order* of the queries to enhance the performance. In the original order in random, the queries do not have the optimal classifications and implementations with  $ELM\_CMR$  model.

Figure 10 shows the performance of the multiple queries about different types under different *execution orders*, respectively, top- $k$  (2 G), top- $k$  (8 G),  $k$ NN (2 G), and  $k$ NN (8 G) in uniform distribution. We can see that the performance under our  $E2E$  model is much optimal than the  $ELM\_CMR$  model. The optimized *execution order* of  $E2E$  model does not only consider the classification results of  $E2E$  model, but also consider the characteristics of the queries deeply.

## 6. Conclusions

In this paper, we propose an efficient query processing optimization model based on ELM,  $E2E$  model. Our  $E2E$  can effectively analyze the query processing applications and then generates the most optimized executions of query processing applications. After analyzing the problems of the former  $ELM\_CMR$  model, we use two stages model to classify the query processing applications in ComMapReduce



FIGURE 10: The different query types.

framework. Then, we propose an efficient training approach to train our model. We also give the query executions based on *E2E* model in two situations, Just-in-Time execution and Queue execution. The experiments demonstrate that the *E2E* model is efficient and the query processing applications can reach an optimal performance.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This research is supported by National Natural Science Foundation of China (nos. 60873068, 61472169, 61100022, and 61472069) and Talent Projects of the Educational Department of Liaoning Province (no. LR201017).

## References

- [1] A. Basu, S. Shuo, H. Zhou, M. Hiot Lim, and G.-B. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, 2013.
- [2] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, 2014.
- [3] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [4] J. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [5] B. P. Chacko, V. R. V. Krishnan, G. Raju, and P. B. Anto, "Handwritten character recognition using wavelet energy and extreme learning machine," *International Journal of Machine Learning and Cybernetics*, vol. 3, no. 2, pp. 149–161, 2012.
- [6] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [7] D. Deng, G. Li, S. Hao, J. Wang, and J. Feng, "MassJoin: a mapreduce-based method for scalable string similarity joins," in *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE '14)*, pp. 340–351, Chicago, Ill, USA, April 2014.
- [8] L. Ding, G. Wang, J. Xin, X. Wang, S. Huang, and R. Zhang, "ComMapReduce: an improvement of MapReduce with lightweight communication mechanisms," *Data and Knowledge Engineering*, vol. 88, pp. 224–247, 2013.
- [9] L. Ding, J. Xin, and G. Wang, "An efficient query processing optimization based on ELM in the cloud," *Neural Computing and Applications*, 2014.
- [10] L. Ding, J. Xin, G. Wang, and S. Huang, "ComMapReduce: an improvement of mapreduce with lightweight communication mechanisms," in *Database Systems for Advanced Applications*, pp. 150–168, Springer, 2012.
- [11] C. Doukeridis and K. Nørvg, "A survey of large-scale analytical query processing in MapReduce," *The VLDB Journal*, vol. 23, no. 3, pp. 355–380, 2014.
- [12] M. Ester and H. Kriegel, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.
- [13] S. Fries, B. Boden, G. Stepien, and T. Seidl, "PHiDJ: parallel similarity self-join for high-dimensional vector data with MapReduce," in *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE '14)*, pp. 796–807, Chicago, Ill, USA, April 2014.
- [14] J. Gao, J. Zhou, C. Zhou, and J. X. Yu, "GLog: a high level graph analysis system using MapReduce," in *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE '14)*, pp. 544–555, IEEE, April 2014.
- [15] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on MapReduce," *Neurocomputing*, vol. 102, pp. 52–58, 2013.
- [16] G. B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.
- [17] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, July 2004.
- [19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [20] W. Jun, W. Shitong, and F.-L. Chung, "Positive and negative fuzzy rule system, extreme learning machine and image classification," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 4, pp. 261–271, 2011.
- [21] Y. Lan, Z. Hu, Y. C. Soh, and G.-B. Huang, "An extreme learning machine approach for speaker recognition," *Neural Computing and Applications*, vol. 22, no. 3–4, pp. 417–425, 2013.
- [22] A. Lendasse, Q. He, Y. Miche, and G.-B. Huang, "Advances in extreme learning machines (ELM2012)," *Neurocomputing*, vol. 128, pp. 1–3, 2014.

- [23] W. Lin, X. Xiao, and G. Ghinita, "Large-scale frequent subgraph mining in MapReduce," in *Proceedings of the 30th IEEE International Conference on Data Engineering (ICDE '14)*, pp. 844–855, April 2014.
- [24] K. Mullesgaard, J. L. Pedersen, H. Lu, and Y. Zhou, "Efficient skyline computation in MapReduce," in *Proceedings of the 17th International Conference on Extending Database Technology (EDBT '14)*, pp. 37–48, 2014.
- [25] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.
- [26] L. Qin, J. X. Yu, L. Chang, H. Cheng, C. Zhang, and X. Liu, "Scalable big graph processing in MapReduce," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*, pp. 827–838, 2014.
- [27] W. Xi-Zhao, S. Qing-Yan, M. Qing, and Z. Jun-Hai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3–9, 2013.
- [28] J.-H. Zhai, H.-Y. Xu, and X.-Z. Wang, "Dynamic ensemble extreme learning machine based on sample entropy," *Soft Computing*, vol. 16, no. 9, pp. 1493–1502, 2012.
- [29] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, and Y. C. Soh, "Dynamic extreme learning machine and its approximation capability," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2054–2065, 2013.
- [30] X. Zhang, L. Chen, and M. Wang, "Efficient multi-way theta-join processing using MapReduce," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1184–1195, 2012.
- [31] W. Zong and G.-B. Huang, "Learning to rank with extreme learning machine," *Neural Processing Letters*, vol. 39, no. 2, pp. 155–166, 2014.

## Research Article

# Distributed Learning over Massive XML Documents in ELM Feature Space

Xin Bi, Xiangguo Zhao, Guoren Wang, Zhen Zhang, and Shuang Chen

College of Information Science and Engineering, Northeastern University, Shenyang, Liaoning 110819, China

Correspondence should be addressed to Xiangguo Zhao; [zhaoxiangguo@mail.neu.edu.cn](mailto:zhaoxiangguo@mail.neu.edu.cn)

Received 21 August 2014; Accepted 16 October 2014

Academic Editor: Tao Chen

Copyright © 2015 Xin Bi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the exponentially increasing volume of XML data, centralized learning solutions are unable to meet the requirements of mining applications with massive training samples. In this paper, a solution to distributed learning over massive XML documents is proposed, which provides distributed conversion of XML documents into representation model in parallel based on MapReduce and a distributed learning component based on Extreme Learning Machine for mining tasks of classification or clustering. Within this framework, training samples are converted from raw XML datasets with better efficiency and information representation ability and taken to distributed learning algorithms in Extreme Learning Machine (ELM) feature space. Extensive experiments are conducted on massive XML documents datasets to verify the effectiveness and efficiency for both classification and clustering applications.

## 1. Introduction

Classification and clustering are two major problems of XML documents mining tasks. One of the most important parts of mining XML documents is to convert XML documents into *representation model*. Most traditional representation models are designed for plain text mining applications, taking no account of structural information of XML documents. Vector Space Model (VSM) [1] is one of the most classic and popular representation models of plain text. Previous work proposed some approaches to solve the problem of considering both semantic and structural information for XML document classification, among which Structured Link Vector Model (SLVM) [2] extends VSM to generate a matrix by recording the attribute values of each element in an XML document. Reduced Structured Vector Space Model (RS-VSM) proposed in [3] achieves a higher performance due to its feature subset selection method. Different weights are assigned to different elements according to their priority and representation ability. Distribution based Structured Vector Model (DSVM) [4] further improves the calculation of traditional Term Frequency Inverse Document Frequency (TFIDF) values and takes two factors into consideration,

namely, Among Classes Discrimination and Within Class Discrimination.

Extreme Learning Machine (ELM) was proposed by Huang et al. in [5, 6] based on generalized single-hidden layer feedforward networks (SLFNs). With its variants [7–11], ELM achieves extremely fast learning capacity and good generalization capabilities usually in many application fields, including text classification [12], multimedia recognition [13–15], bioinformatics [16], and mobile objects [17]. Recently, Huang et al. in [18] pointed out that (1) the maximal margin property of Support Vector Machine (SVM) [19] and the minimal norm of weights theory of ELM are consistent; (2) from the standard optimization method point of view, ELM for classification and SVM are equivalent. Furthermore, it is proved in [20] that (1) ELM provides a unified learning platform with a widespread type of feature mappings; (2) ELM can be implemented in regression and multiclass classification applications in one formula directly. ELM can be linearly extended to SVMs [18] and SVMs can apply ELM kernel to get better performance due to its *universal approximation capability* [10, 11, 21] and *classification capability* [20].

It is generally believed that all the ELM based algorithms consist of two major stages [22]: (1) random feature mapping

and (2) output weights calculation. The first stage generating feature mapping randomly is the key concept in ELM theory which differs from other feature learning algorithms. In view of the good properties of the ELM feature mapping, most existing ELM based classification algorithms can be viewed as *supervised learning in ELM feature space*. While, in [23], the *unsupervised learning in ELM feature space* is studied, drawing the conclusion that the proposed ELM  $k$ -Means algorithm and ELM NMF (nonnegative matrix factorization) clustering can get better clustering results than traditional algorithms in original feature space.

Recently, the volume of XML documents keeps explosively increasing in various kinds of web applications. Since the larger the training sample is, generally the better the learning model will be trained [24], it is a great challenge to implement distributed learning solutions to process massive XML datasets in parallel. MapReduce [25], introduced by Google to process parallelizable problems across huge datasets on clusters of computers, provides tremendous parallel computing power without concerns for the underlying implementation and technology. However, MapReduce framework requires *distributed storage* of the datasets and *no communication* among mappers or reducers, which brings challenges to (1) converting XML datasets into global representation model and (2) implementing learning algorithms in ELM feature space.

To our best knowledge, this paper is the *first* to discuss massive XML documents mining problems. We present a distributed solution to XML representation and learning in ELM feature space. Since the raw XML datasets are stored on distributed file system, we propose algorithm DXRC to convert the XML documents into training samples in the form of XML representation model using a MapReduce job. With the converted training samples, we apply PELM [26] and POS-ELM [27] to realize supervised learning and propose a distributed  $k$ -Means in ELM feature space based on ELM  $k$ -Means proposed in [23]. The contributions can be summarized as follows.

- (1) A *distributed representing algorithm* is proposed to convert massive XML documents into XML representation model in parallel.
- (2) Existing *distributed supervised learning algorithms in ELM feature space* are implemented to make comparison of massive XML documents classification performance, including PELM and POS-ELM.
- (3) A *distributed unsupervised learning algorithm* is proposed based on ELM  $k$ -Means [23] to realize distributed clustering over massive XML documents in *ELM feature space*.
- (4) Empirical and extensive comparison experiments are conducted on clusters to verify the performance of our solution.

The remainder of this paper is structured as follows. Section 2 introduces XML documents representation models and proposes a distributed converting algorithm to represent XML documents stored on distributed file system. Extreme Learning Machine feature mapping is presented in Section 3.

Section 4 presents classification algorithms based on distributed ELMs and proposes a distributed clustering algorithm in ELM feature space based on MapReduce. Section 6 makes performance comparison among distributed classification algorithms and evaluates the proposed distributed clustering algorithm. Section 7 draws conclusions of this paper.

## 2. Distributed XML Representation

In this section, we first introduce representation model of XML documents and then propose a distributed converting algorithm, which is able to generate global feature vectors for all the XML documents stored on distributed file system.

*2.1. XML Representation Model.* For learning problems of texts, such as XML and plain documents, the first important task is to convert original documents into representation model. Vector Space Model (VSM) [1] is often used to represent plain text documents, which takes term occurrence statistics as feature vectors. However, representing an XML document in VSM directly will lose the structural information. Structured Link Vector Model (SLVM) is proposed in [2] based on VSM to represent semistructured documents, which contains both semantic and structural information. SLVM is defined as

$$\mathbf{d}_{\text{slvm}} = \langle \mathbf{d}_1, \dots, \mathbf{d}_n \rangle, \quad (1)$$

where  $\mathbf{d}_i$  is a feature vector of the  $i$ th XML element calculated as

$$\mathbf{d}_i = \sum_j (\text{TF}(w_i, \text{doc}.e_j) \times \varepsilon_j) \text{IDF}(w_i), \quad (2)$$

where  $w_i$  is the  $i$ th term and  $\varepsilon_j$  is a unit vector corresponding to the element  $e_j$ .

In SLVM, each  $\mathbf{d}_{\text{slvm}}$  is a feature matrix  $\mathbf{R}^{n \times m}$ , which is viewed as an array of VSMs.  $\mathbf{d}_i$  consists of the feature terms corresponding to the same XML element, which is an  $m$ -dimensional feature vector in each element unit.

Based on SLVM, in [3], we proposed Reduced Structured Vector Space Model (RS-VSM), which not only inherits the advantages of representing structural information of SVLM, but also achieves a better performance due to the feature subset selection based on information gain. We also proposed Distribution Based Structured Vector Model (DSVM) in [4] to further strengthen the ability of representation. Two improved interact factors were designed, including Among Classes Discrimination (ACD) and Within Class Discrimination (WCD). Revised IDF was also introduced to indicate the importance of a feature term in other classes more precisely.

$\mathbf{d}_{u_i}^k$  is the  $k$ th term feature described as

$$\mathbf{d}_{u_i}^k = \sum_{j=1}^m (\text{TF}(w_k, \text{doc}.e_j) \cdot \varepsilon_j) \cdot \text{IDF}_{\text{ex}}(w_k, c) \cdot \rho_{\text{CD}}, \quad (3)$$

where  $m$  is the number of elements in document doc,  $\text{doc}.e_j$  is the  $j$ th element  $e_j$  of doc, and  $\varepsilon_j$ , which is the unit vector

```

Input: ⟨docID, content⟩
Output: ⟨term, ⟨docID, element, times, sum⟩⟩
(1) Initiate HashMap mapEle;
(2) foreach element ∈ content do
(3)   Initiate sum = 0;
(4)   Initiate HashMap mapEleTF;
(5)   foreach term ∈ element do
(6)     sum++;
(7)     if mapEleTF.containsKey(term) then
(8)       mapEleTF.put(term, mapEleTF.get(term) + 1);
(9)     else
(10)      mapEleTF.put(term, 1);
(11)    mapEle.put(element, mapEleTF);
(12) foreach itrEle ∈ mapEle do
(13)   element = itrEle.getKey();
(14)   foreach itrEleTF ∈ itrEle.getValue() do
(15)     term = itrEleTF.getKey();
(16)     times = itrEleTF.getValue();
(17)     emit(term, ⟨docID, element, times, sum⟩);

```

ALGORITHM 1: Mapper of DXRC.

```

Input: ⟨term, list(⟨docID, element, times, sum⟩)⟩
Output: training samples matrix in the form of ⟨position, tfidf⟩
(1) Initiate HashMap mapDocEleTF;
(2) Initiate HashMap mapTDocs;
(3)  $N = \text{DistributedCache.get}(\text{"totalDocsNum"})$ ;
(4)  $\text{weights} = \text{DistributedCache.get}(\text{"elementWeightsVector"})$ ;
(5) foreach itr ∈ list do
(6)    $\text{weightedDocEleTF} = \text{weights}[\text{docId}, \text{element}] * \text{itr.times}/\text{itr.sum}$ ;
(7)   mapDocEleTF.put(⟨docId, element⟩, weightedDocEleTF);
(8)   if mapTDocs.containsKey(docId) then
(9)      $\text{newTimes} = \text{mapTF.get}(\text{docId}) + \text{irt.getValue().times}$ ;
(10)    mapTDocs.put(docID, newTimes);
(11)   else
(12)    mapTDocs.put(docID, itr.getValue().times);
(13)    $\text{docsNumber} = \text{mapTDocs.size}()$ ;
(14)    $\text{idf} = \log(\text{mapTDocsSize}/N)$ ;
(15) foreach itrDocEleTF ∈ mapDocEleTF do
(16)   position = itrDocEleTF.getKey();
(17)    $\text{tfidf} = \text{itrDocEleTF.getValue}() \times \text{idf}$ ;
(18)   emit(position, tfidf);

```

ALGORITHM 2: Reducer of DXRC.

of  $\text{doc}.e_j$  in SLVM, is now the dot product of  $s$ -dimensional unit vector and  $s$ -dimensional weight vector.  $\text{IDF}_{\text{ex}}(w_i, c)$  is the revised IDF. The factor  $\rho_{\text{CD}}$  is the distribution modifying factor, which equals the reciprocal of arithmetic product of WCD and ACD. The detailed calculation of  $\mathbf{d}_{u_i}^k$  can be found in [4].

**2.2. Distributed Converting Algorithm.** In this section, we propose a distributed converting algorithm, named Distributed XML Representation Converting (DXRC), to calculate TFIDF [28] values of DSVM based on MapReduce. Since the volume of XML documents is so large that the representation model cannot be generated on a single machine, DXRC

realizes the representation of XML documents in the form of DSVM in parallel. The map function and reduce function of DXRC are presented as Algorithms 1 and 2, respectively.

The map function in Algorithm 1 accepts key-value pairs and the MapReduce job context as input. The key of key-value pairs is the XML document ID and the value is the corresponding XML document content. A HashMap *mapEle* (Line 1) is used to cache all the elements of one XML document (Lines 2–11), using element name as key and another HashMap *mapEleTF* (Line 4) as value. The *mapEleTF* caches the TF values of all the words in one element (Lines 5–10). That is, for each XML document, the numbers of *mapEle* items and XML elements are the same;

for each element, there are as many items in  $mapEleTF$  as there are distinct words in this element. Each item in  $mapEle$  and  $mapEle$  will be emitted as output in the form of  $\langle term, \langle docID, element, times, sum \rangle \rangle$  (Lines 12–17).

After the  $\langle term, \langle docID, element, times, sum \rangle \rangle$  pairs are emitted by map function, all the key-value pairs with the same key, which are also the key-value pairs of the same word in XML documents, are combined and passed to the same *reduce* function in Algorithm 2 as input. For each key-value pair processed by reduce function, two HashMaps  $mapDocEleTF$  (Line 1) and  $mapTDocs$  (Line 2) are initiated. The HashMap  $mapDocEleTF$  is to cache the *tf* values of a word in each element in the corresponding XML document and  $mapTDocs$  is to cache the number of documents containing this word. The total number of documents  $N$  (Line 3) and the vector *weights* (Line 4), which indicates the weights of all the elements in each XML document, are obtained through distributed cache defined in MapReduce job configuration. Since reduce now has all the *tf* values grouped by XML elements along with their weights, weighted *tf* values (Line 6) and the number of documents containing each word are calculated and cached in  $mapDocEleTF$  and  $mapTDocs$ , respectively (Lines 5–12). Then the *idf* value can be calculated (Line 14) and multiplied by each item in  $mapDocEleTF$ . The output of reduce is the  $\langle position, tfidf \rangle$  pairs, of which *position* is  $\langle docID, element \rangle$  indicating the index of DSVM matrix and *tfidf* is the value of the matrix. Finally, the XML representation DSVM can be built by this matrix and factor  $\rho_{CD}$ , uploaded onto distributed file system, and used as the input of the training model.

### 3. ELM Feature Mapping

Extreme Learning Machine (ELM) randomly generates parameters of the single-hidden layer feedforward networks without iteratively tuning to gain extremely fast learning speed. The output weights can be calculated by matrix multiplication after the training samples are mapped into *ELM feature space*.

Given  $N$  arbitrary samples  $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^{n \times m}$ , ELM is modeled as

$$\sum_{i=1}^L \beta_i G(\mathbf{w}_i, b_i, \mathbf{x}) = \beta \mathbf{h}(\mathbf{x}), \quad (4)$$

where  $L$  is the number of hidden layer nodes,  $\beta_i$  is the output weight from the  $i$ th hidden node to the output node,  $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$  is the input weight vector, and  $b_i$  is the bias of  $i$ th hidden node.  $G(\mathbf{x})$  is the activation function to generate mapping neurons, which can be any nonlinear piecewise continuous functions [22], including Sigmoid function (5) and Gaussian function (6), as follows:

$$G_{\text{Sigmoid}}(\mathbf{w}, b, \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^T \mathbf{x} + b))}, \quad (5)$$

$$G_{\text{Gaussian}}(\mathbf{w}, b, \mathbf{x}) = \exp(-b \|\mathbf{x} - \mathbf{a}\|). \quad (6)$$

Figure 1 shows the structure of ELM with multiple output nodes and the feature mapping process. The three layers

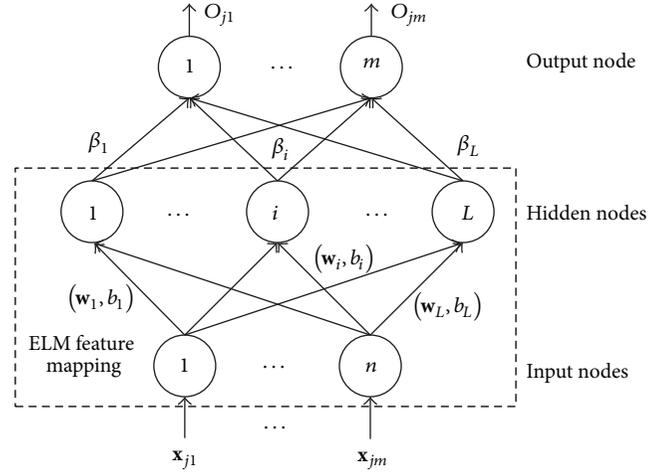


FIGURE 1: ELM structure and ELM feature mapping.

of ELM network are input layer, hidden layer, and output layer. The  $n$  input nodes correspond to the  $n$ -dimensional data space of original samples, while  $L$  hidden nodes correspond to the  $L$ -dimensional *ELM feature space*. With the  $m$ -dimensional output space, the decision function outputs the class label of the samples.

The *ELM feature mapping* denoted by  $\mathbf{H}$  is calculated as

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{x}_1) \\ \vdots \\ h(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\mathbf{w}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{w}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{w}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{w}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}. \quad (7)$$

### 4. Distributed Classification in ELM Feature Space

In this section, we introduce the learning procedure of classification problems in ELM feature space and distributed implementations based on two existing representative distributed ELM algorithms, which are PELM [26] and POS-ELM [27].

**4.1. Supervised Learning in ELM Feature Space.** Most of the existing ELM algorithms aim at supervised learning, that is, classification and regression. In supervised learning applications, ELM is to minimize the training error and the norm of the output weights [5, 6], that is,

$$\text{Minimize: } \|\mathbf{H}\beta - \mathbf{T}\|^2, \|\beta\|, \quad (8)$$

where  $\mathbf{T} = [\mathbf{t}_1^T, \dots, \mathbf{t}_L^T]^T$  is the vector of class labels.

The matrix  $\beta$  is the output weight, which is calculated as

$$\beta = \mathbf{H}^\dagger \mathbf{T}, \quad (9)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose inverse of  $\mathbf{H}$ .

ELM for classification is presented as Algorithm 3.

The output weight of ELM can also be calculated as

$$\beta = \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}, \quad (10)$$

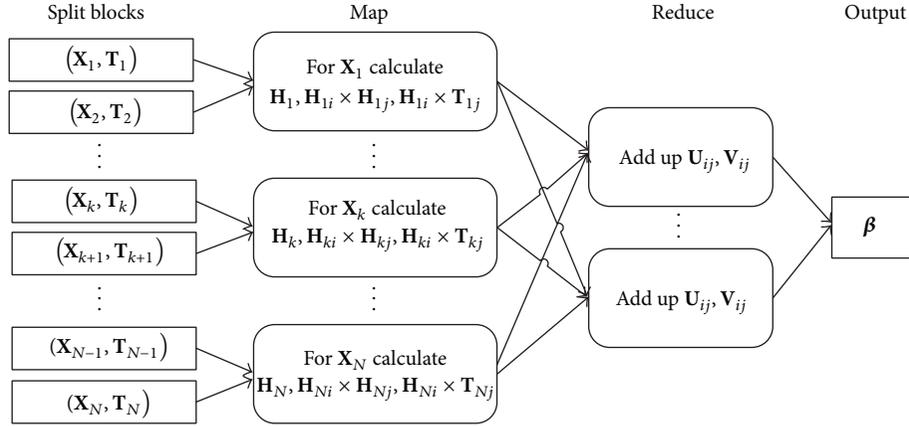


FIGURE 2: Parallel ELM.

- (1) **for**  $i = 1$  to  $L$  **do**
  - (2) Randomly assign input weight  $\mathbf{w}_i$  and bias  $b_i$ ;
  - (3) Calculate ELM feature space  $\mathbf{H}$ ;
  - (4) Calculate  $\beta = \mathbf{H}^T \mathbf{T}$ ;

ALGORITHM 3: Extreme Learning Machine for classification.

where, according to the ridge regression theory [29], the diagonal of a symmetric matrix can be incremented by a biasing constant  $1/C$  to gain better stability and generalization performance [18].

For the case in which the number of training samples is much larger than the dimensionality of the feature space, considering the computation cost, the output weight calculation equation can be rewritten as

$$\beta = \left( \frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}. \quad (11)$$

**4.2. Distributed Implementations.** Some existing works have introduced distributed implementations of various ELM algorithms. The original ELM was parallelized by PELM in [26]; Online Sequential ELM (OS-ELM) was implemented on MapReduce as POS-ELM in [27].

**4.2.1. Parallel ELM.** In the original ELM algorithm, in the case that the number of training samples is much larger than the dimensionality of ELM feature space and  $\mathbf{H}^T \mathbf{H}$  is nonsingular, the major cost is the calculation of Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ , where the orthogonal projection method is used as (11). Thus the matrix multiplication  $\mathbf{U} = \mathbf{H}^T \mathbf{H}$  and  $\mathbf{V} = \mathbf{H}^T \mathbf{T}$  can be calculated

by a MapReduce job. In map function, each term of  $\mathbf{U}$  and  $\mathbf{V}$  can be expressed as follows [26]:

$$\mathbf{U}[i][j] = \sum_{k=1}^N \mathbf{H}[k][i] \times \mathbf{H}[k][j], \quad (12)$$

$$\mathbf{V}[i][j] = \sum_{k=1}^N \mathbf{H}[k][i] \times \mathbf{T}[k][j].$$

In reduce function, all the intermediate results are merged and added up according to the corresponding elements of the result matrix. Since the training input matrix  $\mathbf{X}$  is stored by sample on different machines, the calculation can be parallelized and executed by the MapReduce job. The calculation procedure is demonstrated as Figure 2.

**4.2.2. Parallel Online Sequential ELM.** The basic idea of Parallel Online Sequential ELM (POS-ELM) is to calculate  $\mathbf{H}_1, \dots, \mathbf{H}_B$  in parallel. By taking advantages of the calculation of partial ELM feature matrix  $\mathbf{H}_i$  with a chunk of training data of OS-ELM, POS-ELM calculates its  $\mathbf{H}_i$  with its own data chunk in the map phase on each machine. The reduce function collects all the  $\mathbf{H}_i$  and calculates  $\beta_{k+1}$  as

$$\beta_{k+1} = \beta_k + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \beta_k), \quad (13)$$

where

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k. \quad (14)$$

The calculation procedure of POS-ELM is shown in Figure 3.

## 5. Distributed Clustering in ELM Feature Space

In this section, in order to improve the efficiency of clustering massive XML datasets, we also propose a parallel implementation of ELM  $k$ -Means algorithm, named Distributed ELM  $k$ -Means (DEK) in this section.

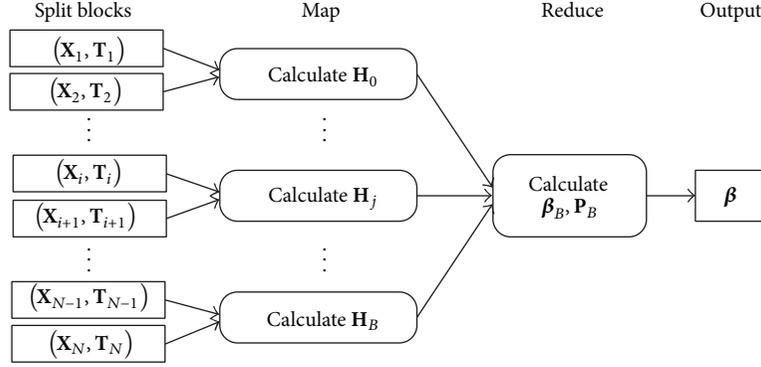
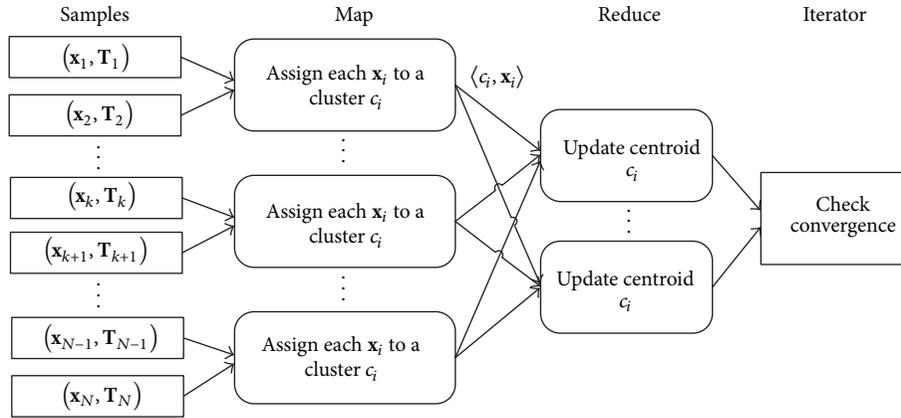


FIGURE 3: Parallel Online Sequential ELM.

FIGURE 4: Distributed ELM  $k$ -Means.

**5.1. Unsupervised Learning in ELM Feature Space.** It is believed that transforming nonlinear data into some high dimensional feature space increases the probability of the linear separability. However, many Mercer kernel based clustering algorithms are usually not efficient for computation, since the feature mapping is always implicit and cannot be guaranteed to satisfy the universal approximation condition. Thus, [23] holds that explicit feature mapping like ELM feature mapping is more appropriate.

Generally,  $k$ -Means algorithm in ELM feature space, as ELM  $k$ -Means for short, has two major steps: (1) transform the original data into ELM feature space and (2) implement traditional clustering algorithm directly. Clustering in the ELM feature space is much more convenient than kernel based algorithms.

**5.2. Distributed ELM  $k$ -Means.** For the applications of massive XML documents clustering, implementation of unsupervised learning methods to MapReduce is a key part of the problem. Since ELM feature mapping is extremely fast with good generalization performance and universal approximation ability, in this section, we propose Distributed ELM  $k$ -Means (DEK) based on ELM  $k$ -Means [23].

In DEK algorithm, the training samples of XML documents are distributedly stored on distributed file system. Each

$(\mathbf{x}_i, T_i)$  represents a training sample  $\mathbf{x}_i$  in ELM feature space with its corresponding class label  $T_i$ . With a set of initiated  $k$  cluster centroids, in the *map* phase, the distances between each centroid and each training sample stored on its own site are calculated. Then the sample is assigned to the centroid with the shortest distance. In the *reduce* phase, all the samples assigned to the same centroid are collected in the same reducer. Then a new centroid of each cluster is calculated, with which the set of  $k$  cluster centroids are updated. That is, a round of MapReduce job updates the set of  $k$  cluster centroids once. In the next round of MapReduce job, the updated set of centroids is updated again and this procedure will be repeated until convergence or up to maximum number of iterations (Figure 4).

Algorithm 4 presents the map function of DEK. For each sample stored on this mapper (Line 1), the distance between the sample and each cluster centroids is calculated (Lines 2, 3). Then each sample is assigned to the cluster whose centroid is the nearest to this sample (Line 4). The intermediate key-value pair is emitted in the form of  $\langle c_{\max}, \mathbf{x}_i \rangle$  (Line 5), in which  $\mathbf{x}_i$  is the specific sample and  $c_{\max}$  is the assigned cluster of  $\mathbf{x}_i$ .

Algorithm 5 presents the reduce function of DEK. We add up the sum distance in Euclidean space of all the samples  $\mathbf{x}_i$  in list(x) (Lines 1, 2) and then calculate the mean value to represent the new version of the centroid  $c_j^{\text{updated}}$  of cluster

**Input:** Training samples  $X$ ,  $k$  centroids  $C$

**Output:**  $\langle$ centroid  $c_{\max}$ , sample  $x_i$  $\rangle$

- ```

(1) foreach  $x_i \in X$  do
(2)   foreach  $c_j \in C$  do
(3)     Calculate distance  $d_{ij}$  between  $x_i$  and  $c_j$ ;
(4)     Assign  $x_i$  to the cluster  $c_{\max}$  with  $\max_j(d_{ij})$ ;
(5)     Emit  $\langle c_{\max}, x_i \rangle$ ;

```

ALGORITHM 4: Mapper of DEK.

$c_j$  (Line 3). When all the  $k$  cluster centroids are updated in this MapReduce job, if this version of centroids is the same as the older one, or if the maximum number of iterations is reached, DEK holds that the clustering job is done; otherwise, DEK continues to the next iteration of MapReduce job until convergence.

## 6. Performance Evaluation

All the experiments are conducted to compare the performance in the following aspects:

- (i) *scalability* evaluation of proposed Distributed XML Representation Converting (DXRC),
- (ii) *scalability* comparison between PELM and POS-ELM in massive XML documents classification problem,
- (iii) *supervised learning performance* of PELM and POS-ELM in massive XML documents classification problem,
- (iv) *scalability* evaluation of proposed Distributed ELM  $k$ -Means (DEK),
- (v) *unsupervised learning performance* of DEK in massive XML documents clustering problem.

### 6.1. Experiments Setup

**6.1.1. Environment.** All the experiments on distributed XML representation converting and distributed learning in ELM feature space are conducted on a Hadoop (Apache Hadoop, <http://hadoop.apache.org/>) cluster of nine machines, which consists of one master node and eight slave nodes. Each machine is equipped with an Intel Quad Core 2.66 GHZ CPU, 4 GB of memory, and CentOS 5.6 as operating system. All the computers are connected via a high speed Gigabit network. The MapReduce framework is configured with Hadoop version 0.20.2 and Java version 1.6.0\_24.

**6.1.2. Datasets.** Three datasets of XML documents are used as original datasets, which are *Wikipedia XML Corpus* provided by INEX, *IBM DeveloperWorks* (<http://www.ibm.com/developerworks/>) articles, and *ABC News* (<http://abcnews.go.com/>). *Wikipedia XML Corpus* is composed of around 96,000 XML documents classified into 21 classes. We also fetched RSS feeds of news and articles in the format of XML from *IBM DeveloperWorks* and *ABC News* official web sites. Each XML document of the RSS feeds is composed of elements of title, author, summary, publish information, and so forth. In order

to compare the performance of the algorithms over different datasets, we choose the same numbers of XML documents out of all the three datasets, which are 6 classes and 500 documents in each class.

**6.1.3. Parameters.** According to the universal approximation conditions and classification capability of ELM, a large number of hidden nodes guarantee that the data can be linearly separated [23], especially for the learning problems on high-dimensional training samples like XML documents. Thus, after a set of experiments for parameter setting, the only parameter of learning algorithms in ELM feature space, that is, the number of hidden nodes  $L$ , is set to 800.

**6.1.4. Evaluation Criteria.** To clearly evaluate the performance, three sets of evaluation criteria are utilized.

- (1) For scalability evaluation, we compare the criteria of speedup, sizeup, and scaleup. *Speedup* indicates the scalability when increasing the number of running machines, which is measured as

$$\text{Speedup}(m) = \frac{\text{running time on 1 machine}}{\text{running time on } m \text{ machines}}. \quad (15)$$

*Sizeup* indicates the scalability when increasing the data size, which is measured as

$$\text{Sizeup}(m) = \frac{\text{running time over } m \text{ units of data}}{\text{running time over one unit of data}}. \quad (16)$$

*Scaleup* is to measure the scalability of processing  $m$ -times larger data on an  $m$ -times larger cluster, which is calculated as

$$\begin{aligned} \text{Scaleup}(m) \\ = \frac{\text{running time over one unit of data on one machine}}{\text{running time over } m \text{ units of data on } m \text{ machines}}. \end{aligned} \quad (17)$$

- (2) For classification problems, accuracy, recall, and  $F$ -measure are used to evaluate the performance of supervised learning performance in ELM feature space. *Accuracy* indicates the overall ratio of correctly classified samples, which is measured as

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total samples number}}. \quad (18)$$

*Recall* is the ratio of the samples with a specific class label to the ones classified into this class, which is measured as

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False}}. \quad (19)$$

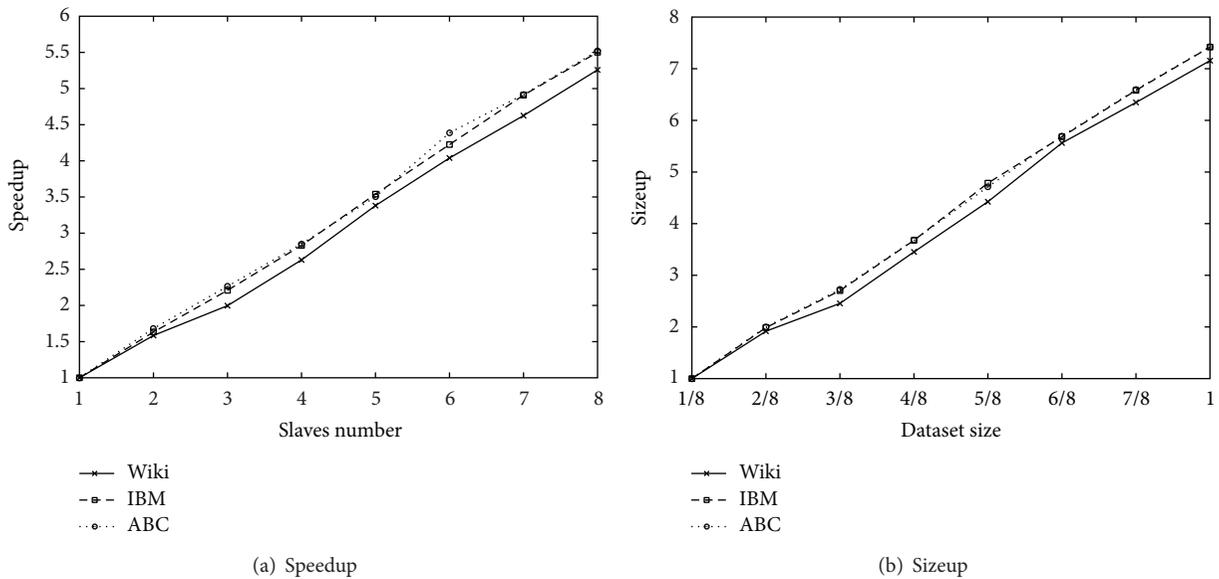
*F-measure* is to measure the overall performance considering both accuracy and recall, which is calculated as

$$F\text{-measure} = \frac{2 \times \text{accuracy} \times \text{recall}}{\text{accuracy} + \text{recall}}. \quad (20)$$

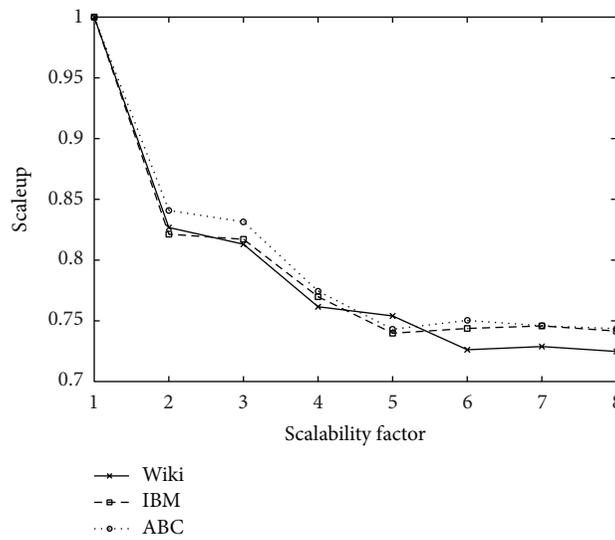
```

Input: (centroid  $c_j$ , samples list( $\mathbf{x}$ ))
Output: Updated set of centroids  $C_{\text{updated}}$ 
(1) foreach  $x_i \in \text{list}(\mathbf{x})$  do
    (2) Add  $x_i$  to squared sum  $S$ ;
    (3) Calculate  $c_j^{\text{updated}}$  of cluster  $c_j$  as  $c_j^{\text{updated}} = S/\text{list}(\mathbf{x}).\text{length}$ ;
    
```

ALGORITHM 5: Reducer of DEK.



(a) Speedup (b) Sizeup



(c) Scaleup

FIGURE 5: Scalability of DXRC.

(3) For clustering problems, since each sample in the datasets we used in our experiments is assigned with a class label, we treat this class label as the cluster label. Thus, the same evaluation criteria are used for clustering problems as for classification problems.

## 6.2. Evaluation Results

6.2.1. Scalability of DXRC. The scalability of representation converting algorithm DXRC is first evaluated. Figure 5(a) demonstrates the speedup of DXRC. As the number of

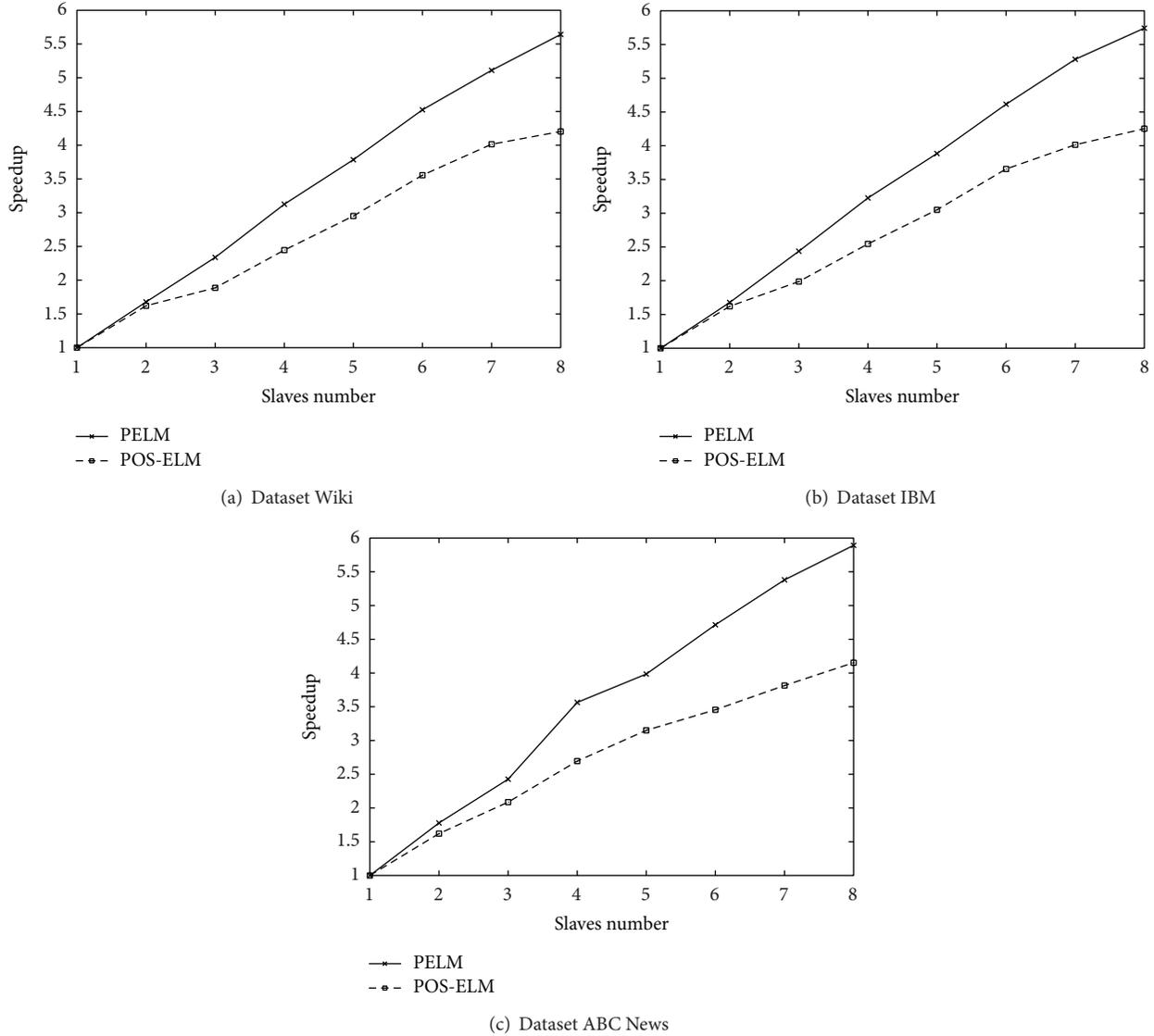


FIGURE 6: Comparison of speedup between PELM and POS-ELM.

the slave nodes varies from one to eight, the speedup tends to be approximately linear at first, but the growth slows down due to the increasing cost of network communications among more and more working machines. But, in general, DXRC gains good speedup. Figure 5(b) presents the sizeup of DXRC. The  $x$ -axis denotes the percentage against the whole datasets. That is, 1 is the full size of original dataset; 0.5 indicates half of the original dataset, in which the samples are randomly chosen. With a fixed number of slave machines, which is eight, the evaluation result shows good sizeup of DXRC. Since the scaleup in distributed implementation cannot stick to 1 in practice, in Figure 5(c), the scaleup of DXRC drops slowly when the number of slave nodes and the size of dataset increase, which indicates a good scaleup of DXRC.

Note that the representation ability and performance influence on XML documents classification of DSVM applied in DXRC can be found in our previous work [4].

6.2.2. Scalability of Massive XML Classification in ELM Feature Space. With the training samples converted by algorithm DXRC, a classifier of massive XML documents can be trained based on MapReduce. The speedup comparison between PELM and POS-ELM on three datasets is presented in Figure 6.

Algorithm PELM, which implements original ELM on MapReduce, requires calculating the inverse of ELM feature space matrix, while PEO-ELM makes use of the idea of online sequential processing to realize parallel computation without communication and requires calculating output weight  $\beta$  and the auxiliary matrix  $\mathbf{P}$  iteratively in a single reducer. The centralized calculation reduces the scalability of both PELM and POS-ELM to some degree, especially for POS-ELM. Thus, the speedup of PELM is better than POS-ELM.

Figure 7 demonstrates the sizeup comparison between PELM and POS-ELM. From this figure, we find that

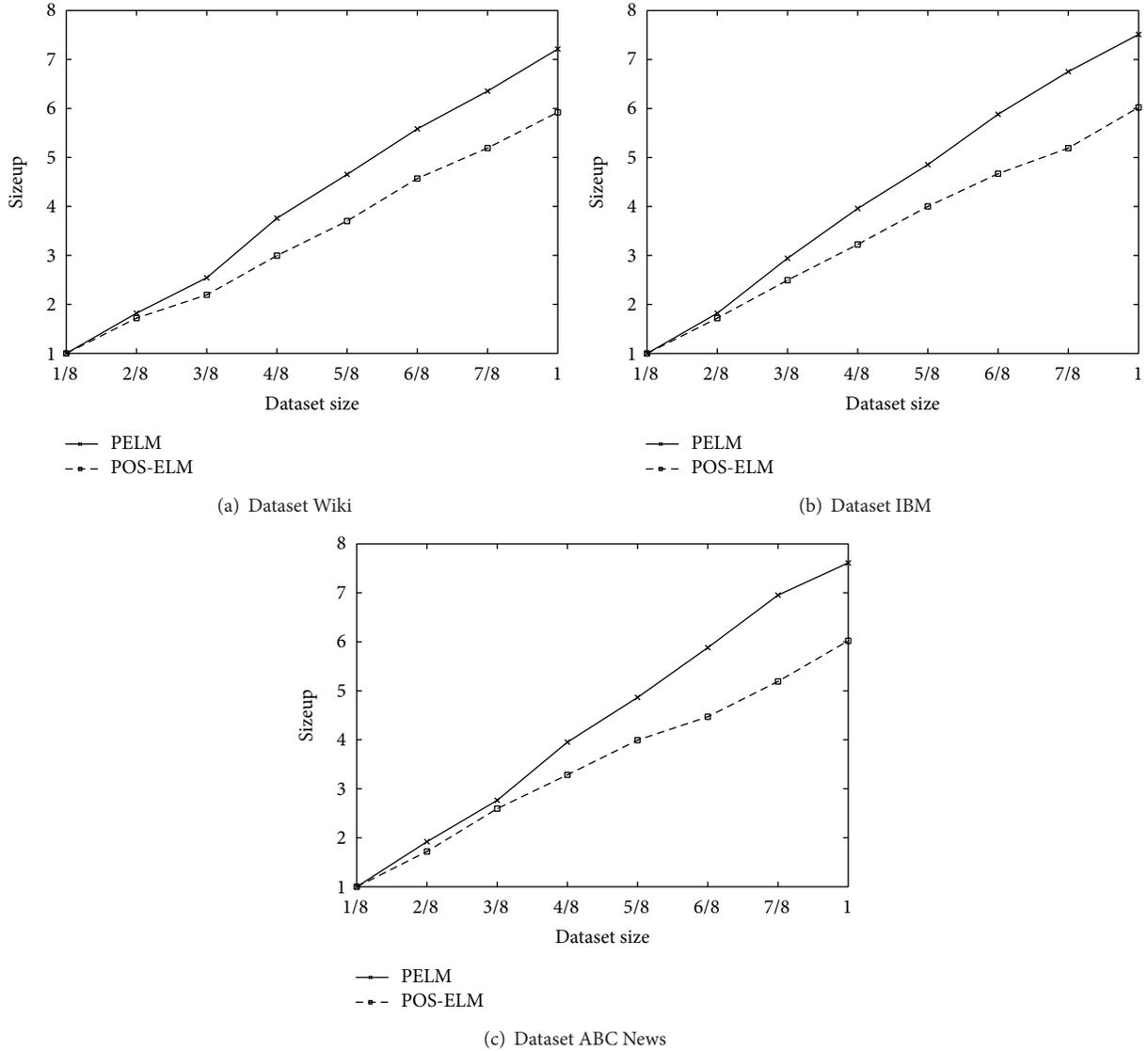


FIGURE 7: Comparison of sizeup between PELM and POS-ELM.

the sizeup of PELM is better than POS-ELM on all the three datasets.

For the scaleup comparison, Figure 8 demonstrates that both PELM and POS-ELM have good scaleup performance, and PELM outperforms POS-ELM on each of the three datasets.

In summary, both PELM and POS-ELM have good scalability for massive XML documents classification applications, but PELM has better scalability than POS-ELM.

**6.2.3. Performance of Massive XML Classification in ELM Feature Space.** The parallel implementation of both PELM and POS-ELM does not invade the computation theory of original ELM and OS-ELM, respectively; that is, the classification performances of PELM and POS-ELM are nearly the same as their corresponding centralized algorithms, respectively. The classification results are shown in Table 1.

From the table we can see that PELM slightly outperforms POS-ELM, because the iterative matrix operations of output weight  $\beta$  in POS-ELM cause loss of calculation accuracy. However, for massive XML documents classification applications, since both the extraction and reduction of XML document features are complicated, both PELM and POS-ELM provide satisfactory classification performance.

**6.2.4. Scalability of Massive XML Clustering in ELM Feature Space.** In this set of experiments, we evaluate the proposed distributed clustering algorithm in ELM feature space, that is, distributed ELM  $k$ -Means. In theory, the scalability of distributed  $k$ -Means in ELM feature space and in original feature space is the same, since the only difference is the feature space of the training samples, which has no influence on the computation complexity. Thus, we only present the scalability of DEK without comparison with distributed  $k$ -Means in original feature space.

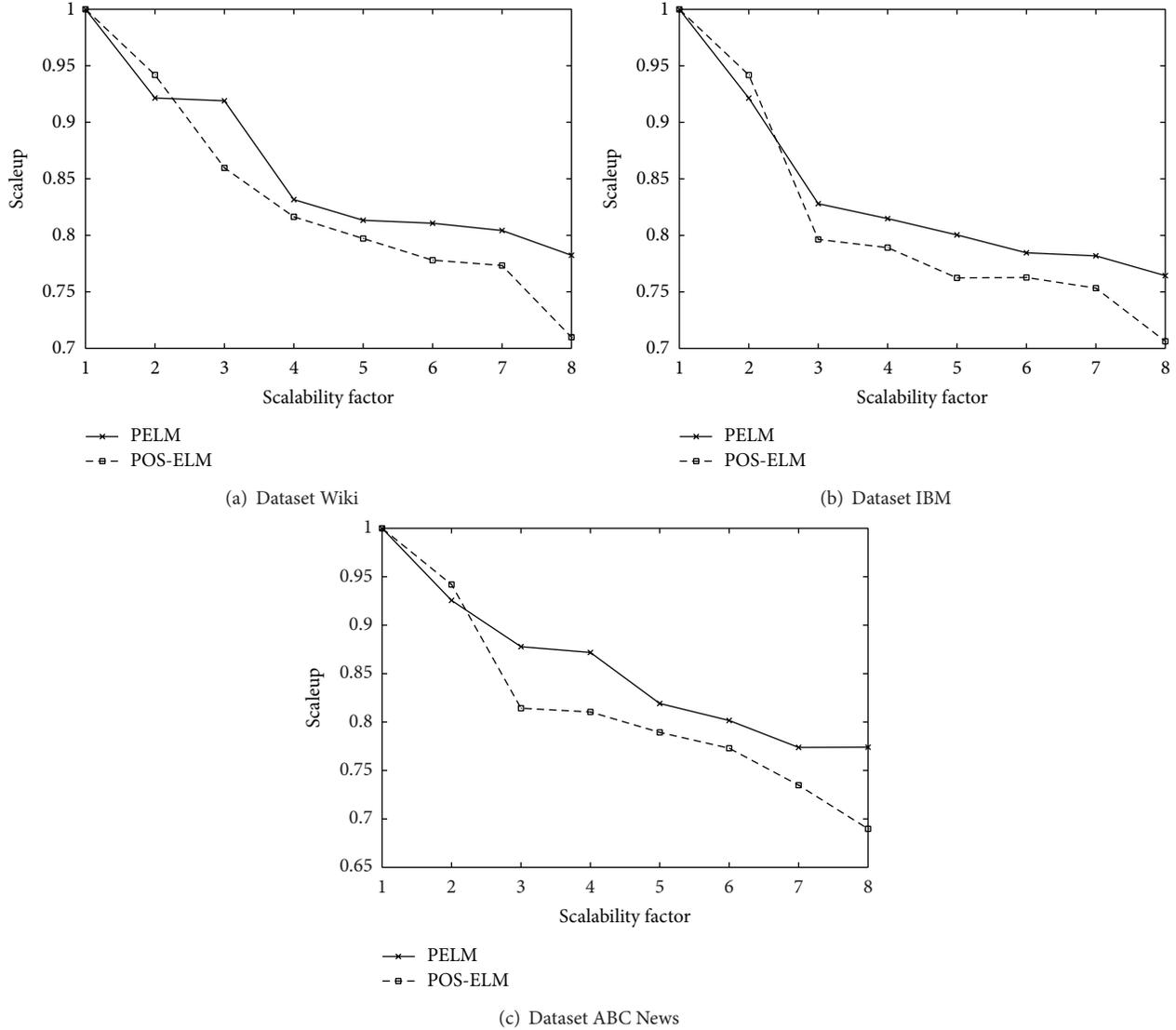


FIGURE 8: Comparison of scaleup between PELM and POS-ELM.

TABLE 1: Classification performance comparison between PELM and POS-ELM.

| Datasets           | PELM     |        |                   | POS-ELM  |        |                   |
|--------------------|----------|--------|-------------------|----------|--------|-------------------|
|                    | Accuracy | Recall | <i>F</i> -measure | Accuracy | Recall | <i>F</i> -measure |
| Wikipedia          | 0.7886   | 0.7563 | 0.7721            | 0.7923   | 0.7745 | 0.7833            |
| IBM DeveloperWorks | 0.7705   | 0.8145 | 0.7919            | 0.7711   | 0.7863 | 0.7891            |
| ABC News           | 0.8681   | 0.8517 | 0.8598            | 0.8517   | 0.8335 | 0.8425            |

The scalability of DEK is evaluated on all the three datasets in terms of speedup in Figure 9(a), sizeup in Figure 9(b), and scaleup in Figure 9(c). The experimental results all demonstrate good scalability of DEK for massive XML documents clustering applications.

6.2.5. Performance of Massive XML Clustering in ELM Feature Space. Clustering performance comparison between distributed clustering in ELM feature space and clustering in original feature space is made for massive XML documents clustering applications in this set of experiments. Note that, since the manual relabeling of the massive XML dataset is

infeasible, we only evaluate the clustering quality with the original number of classes, which is six. The comparison results on three different datasets are presented in Table 2. It can be seen from the comparison results that DEK gets better clustering performance due to its ELM features mapping.

## 7. Conclusion

This paper addresses the problem of distributed XML documents learning in ELM feature space, which has no previous work to our best knowledge. Parallel XML documents representation converting problem based on MapReduce is

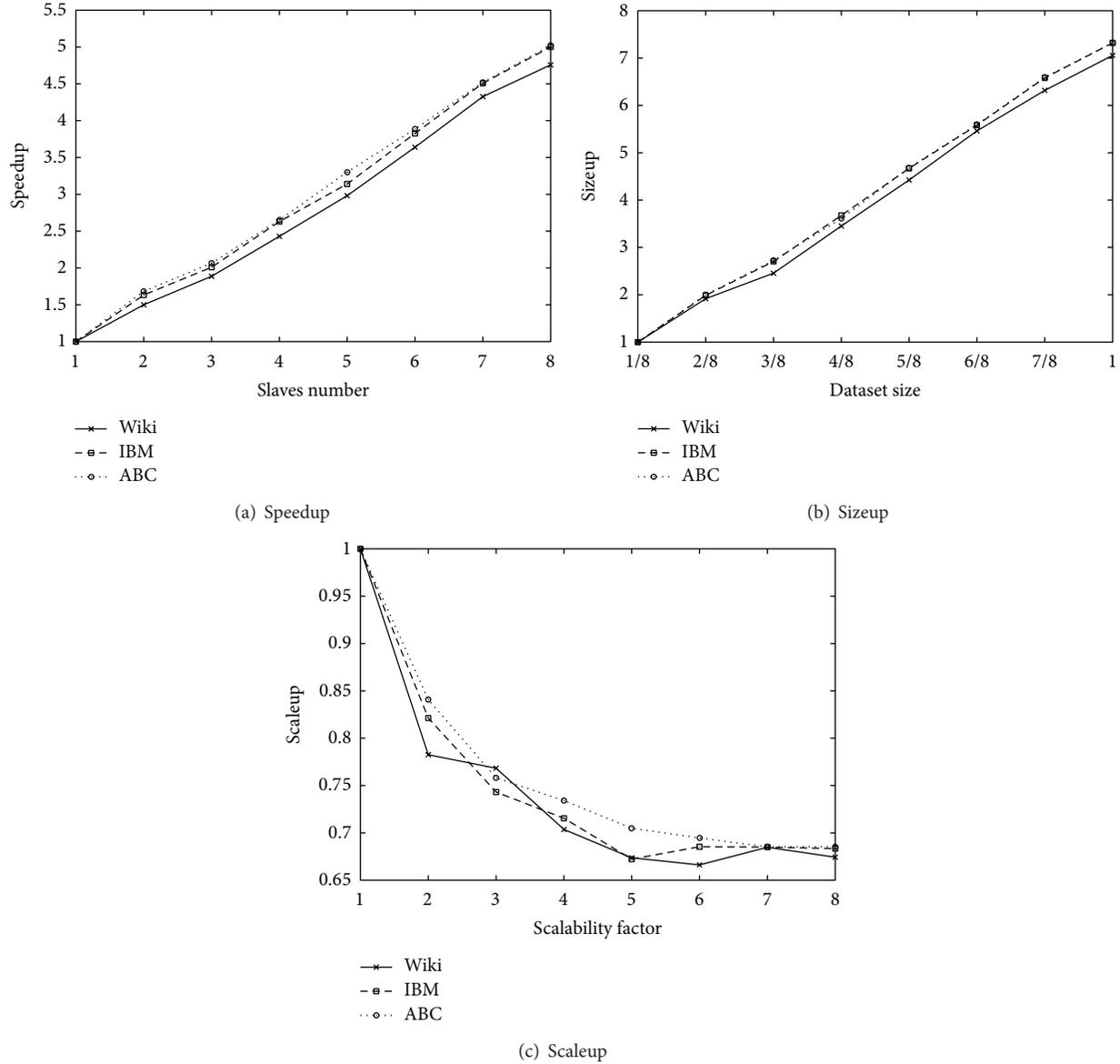


FIGURE 9: Scalability of DEK.

TABLE 2: Clustering performance of DEK compared with parallel  $k$ -Means.

| Dataset            | Parallel $k$ -Means |        |              | DEK      |        |              |
|--------------------|---------------------|--------|--------------|----------|--------|--------------|
|                    | Accuracy            | Recall | $F$ -measure | Accuracy | Recall | $F$ -measure |
| Wikipedia          | 0.7602              | 0.7426 | 0.7513       | 0.7737   | 0.7648 | 0.7692       |
| IBM DeveloperWorks | 0.7985              | 0.8268 | 0.8126       | 0.8124   | 0.8277 | 0.8200       |
| ABC News           | 0.8351              | 0.8125 | 0.8201       | 0.8529   | 0.8192 | 0.8357       |

discussed by proposing a distributed XML representation converting algorithm DXRC. The problem of massive XML documents classification in ELM feature space is studied by implementing PELM and POS-ELM, while, for the problem of massive XML documents clustering in ELM feature space, a distributed ELM  $k$ -Means algorithm DEK is proposed. Experimental results demonstrate that the distributed XML

learning in ELM feature space shows good scalability and learning performance.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This research is partially supported by the National Natural Science Foundation of China under Grants nos. 61272181 and 61173030, the National Basic Research Program of China under Grant no. 2011CB302200-G, the 863 Program under Grant no. 2012AA011004, and the Fundamental Research Funds for the Central Universities under Grant no. N120404006.

## References

- [1] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, NY, USA, 1984.
- [2] J. Yang and X. Chen, "A semi-structured document model for text mining," *Journal of Computer Science and Technology*, vol. 17, no. 5, pp. 603–610, 2002.
- [3] X.-G. Zhao, G. Wang, X. Bi, P. Gong, and Y. Zhao, "XML document classification based on ELM," *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, 2011.
- [4] X. Zhao, X. Bi, and B. Qiao, "Probability based voting extreme learning machine for multiclass XML documents classification," *World Wide Web*, pp. 1–15, 2013.
- [5] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, July 2004.
- [6] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [7] G.-B. Huang, Q.-Y. Zhu, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?" *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 3, pp. 187–191, 2006.
- [8] G. Feng, G.-B. Huang, Q. Lin, and R. K. L. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352–1357, 2009.
- [9] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.
- [10] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.
- [11] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.
- [12] X.-G. Zhao, G. Wang, X. Bi, P. Gong, and Y. Zhao, "XML document classification based on ELM," *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, 2011.
- [13] B. Lu, G. Wang, Y. Yuan, and D. Han, "Semantic concept detection for video based on extreme learning machine," *Neurocomputing*, vol. 102, pp. 176–183, 2013.
- [14] Y. Lan, Z. Hu, Y. C. Soh, and G.-B. Huang, "An extreme learning machine approach for speaker recognition," *Neural Computing and Applications*, vol. 22, no. 3–4, pp. 417–425, 2013.
- [15] W. Zong and G.-B. Huang, "Face recognition based on extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2541–2551, 2011.
- [16] G. Wang, Y. Zhao, and D. Wang, "A protein secondary structure prediction framework based on the extreme learning machine," *Neurocomputing*, vol. 72, no. 1–3, pp. 262–268, 2008.
- [17] B. Wang, G. Wang, J. Li, and B. Wang, "Update strategy based on region classification using ELM for mobile object index," *Soft Computing*, vol. 16, no. 9, pp. 1607–1615, 2012.
- [18] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.
- [19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [20] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [21] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [22] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, 2014.
- [23] Q. He, X. Jin, C. Du, F. Zhuang, and Z. Shi, "Clustering in extreme learning machine feature space," *Neurocomputing*, vol. 128, pp. 88–95, 2014.
- [24] T. Ngo, "Data mining: practical machine learning tools and technique," in *ACM Sigsoft Software Engineering Notes*, I. H. Witten, E. Frank, and M. A. Hell, Eds., pp. 51–52, 3rd edition, 2011.
- [25] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *Operating Systems Design and Implementation*, pp. 137–150, 2004.
- [26] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on mapReduce," *Neurocomputing*, vol. 102, pp. 52–58, 2013.
- [27] B. Wang, S. Huang, J. Qiu, Y. Liu, and G. Wang, "Parallel online sequential extreme learning machine based on MapReduce," *Neurocomputing*, 2014.
- [28] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
- [29] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

## Research Article

# Multiclass AdaBoost ELM and Its Application in LBP Based Face Recognition

Yunliang Jiang,<sup>1,2</sup> Yefeng Shen,<sup>1,3</sup> Yong Liu,<sup>1</sup> and Weicong Liu<sup>1</sup>

<sup>1</sup>*Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China*

<sup>2</sup>*School of Information & Engineering, Huzhou Teachers College, Huzhou 313000, China*

<sup>3</sup>*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China*

Correspondence should be addressed to Yong Liu; [yongliu@iipc.zju.edu.cn](mailto:yongliu@iipc.zju.edu.cn)

Received 22 August 2014; Revised 11 November 2014; Accepted 18 November 2014

Academic Editor: Jiuwen Cao

Copyright © 2015 Yunliang Jiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Extreme learning machine (ELM) is a competitive machine learning technique, which is simple in theory and fast in implementation; it can identify faults quickly and precisely as compared with traditional identification techniques such as support vector machines (SVM). As verified by the simulation results, ELM tends to have better scalability and can achieve much better generalization performance and much faster learning speed compared with traditional SVM. In this paper, we introduce a multiclass AdaBoost based ELM ensemble method. In our approach, the ELM algorithm is selected as the basic ensemble predictor due to its rapid speed and good performance. Compared with the existing boosting ELM algorithm, our algorithm can be directly used in multiclass classification problem. We also carried out comparable experiments with face recognition datasets. The experimental results show that the proposed algorithm can not only make the predicting result more stable, but also achieve better generalization performance.

## 1. Introduction

Many research works have been done in feedforward neural networks, which pointed out that the feedforward neural networks are able to not only approximate complex nonlinear mapping, but also provide models for some natural and artificial problems which classic parametric technics are unable to handle.

Recently, Huang et al. [1] proposed a new simple algorithm based on single layer feedforward networks (SLFNs) called extreme learning machine (ELM). For ELM randomly generates parameters of the networks, its learning speed can be thousands of times faster than traditional feedforward network learning algorithms like back-propagation (BP) algorithm, which needs to iterate many times to get optimal parameters.

In addition, Huang [2] also shows that in theory ELMs (with the same kernels) tend to outperform SVM and its variants in both regression and classification applications with much easier implementation. Based on this conclusion,

the paper in the literature proposed by Wong et al. [3] explores the superiority of the fault identification time of ELM.

In view of the advantages of the algorithm, Cao et al. put it into some areas, such as landmark recognition [4] and protein sequence classification [5]. Besides, Cao et al. [6] proposed an improved learning algorithm which incorporates the voting method into the popular extreme learning machine in classification applications and outperforms the original ELM algorithm as well as several recent classification algorithms.

AdaBoost [7] is one of the most popular algorithms of classifier ensemble to improve the generalization performance. Wang and Li in [8] proposed an algorithm named dynamic AdaBoost ensemble ELM (named DAEELM in this paper). The proposed algorithm takes the ELM as the basic classifier and applies AdaBoost to solve binary classification problem. Similarly, Tian and Mao in [9] combined the modified AdaBoost.RT [10] with ELM to propose a new hybrid artificial intelligent technique called ensemble ELM.

Ensemble ELM aims to improve ELM's performance in regression problem.

However, until now, not so much works have been done to apply AdaBoost to ELM for multiclass classification problem directly. In Freund and Schapire's work [11], they give two extensions of their boosting algorithm to multiclass prediction problems in which each example belongs to one of several possible classes (rather than just two). Since ELM can directly work for multiclass classification problem, this paper proposes an algorithm named multiclass AdaBoost ELM (MAELM). This new algorithm applies multiclass AdaBoost as an ensemble method to a number of ELMs. In addition, this paper proposes a structure to apply ELM and MAELM to local binary patterns (LBP) [12] based face recognition problem. Experiments in LBP based face recognition will show that the proposed algorithm outperforms the original ELM.

This paper is an extension of our previous work [13]. In this paper, we extend our previous work by proposing a new way to combine ELM with PCA instead of using random weights between the input layer and the hidden layer, as well as the bias of the activation function. Experiments in LBP based face recognition will show the stable and good performance with our extended approach.

The rest of the paper is organized as follows. Section 2 gives a brief review of the ELM and PCA, original and multiclass AdaBoost and LBP. The proposed MAELM is presented in Section 3. The experimental result will be shown in Section 4 and a short discussion about the proposed algorithm will be presented in Section 5. Finally, in Section 6, we conclude the paper.

## 2. A Review of Related Work

In this section, a review of the original ELM algorithm and PCA and multiclass AdaBoost and the LBP based face recognition is presented.

*2.1. ELM.* For  $N$  arbitrary distinct samples  $(x_i, t_i)$ , where  $x_i = [x_{i1}, x_{i2}, \dots, x_{id}]^T \in R^d$  and  $t_i = [t_{i1}, t_{i2}, \dots, t_{iK}]^T \in R^K$ , standard SLFNs with  $L$  hidden nodes and activation function  $h(x)$  are mathematically modeled as follows:

$$\sum_{i=1}^L \beta_i h_i(x_j) = \sum_{i=1}^L \beta_i h_i(w_i \cdot x_j + b_i) = o_j, \quad (1)$$

where  $j = 1, 2, \dots, N$ .

Here,  $w_i = [w_{i1}, w_{i2}, \dots, w_{id}]^T$  is the weight vector connecting the  $i$ th hidden node and the input nodes,  $\beta_i = [\beta_{i1}, \dots, \beta_{iK}]^T$  is the weight vector connecting the  $i$ th hidden node and the output nodes, and  $b_i$  is the threshold of the  $i$ th hidden node.

The standard SLFNs with  $L$  hidden nodes with activation function  $h(x)$  can be compactly written as follows:

$$H\beta = T, \quad (2)$$

where

$$H = \begin{bmatrix} h_1(w_1 \cdot x_1 + b_1) & \cdots & h_L(w_L \cdot x_1 + b_L) \\ \vdots & & \vdots \\ h_1(w_1 \cdot x_N + b_1) & \cdots & h_L(w_L \cdot x_N + b_L) \end{bmatrix}, \quad (3)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}.$$

Different from the conventional gradient-based solution of SLFNs, ELM simply solves the function by

$$\beta = H^+ T. \quad (4)$$

$H^+$  is the Moore-Penrose generalized inverse of matrix  $H$ . As Huang et al. have pointed out in [14],  $H^+$  can be represented by

$$H^+ = H^T \left( \frac{I}{C} + HH^T \right)^{-1}, \quad (5)$$

where  $I$  is an identity matrix, which has the same dimension with  $HH^T$ .  $C$  is a constant number which can be set by the user. Adding  $I/C$  can avoid the situation that  $HH^T$  is singular. Huang et al. [1] successfully applied ELM to solve binary classification problem and Huang et al. [14] extended the ELM to directly solve the multiclass classification problem.

Since the original ELM randomly generates the weights between the input layer and the hidden layer, as well as the bias of the activation function, its performance may be not so stable. Instead of that, some other ways like PCA algorithm rewards to try.

*2.2. PCA.* Principal component analysis (PCA) was invented in 1901 by Pearson [15], as an analogue of the principal axes theorem in mechanics, which was later independently developed (and named) by Hotelling in the 1930s [16]. Now, it is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using  $Z$ -scores) the data matrix for each attribute [17]. The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point) and loadings (the weight by which each standardized original variable should be multiplied to get the component score).

The procedure of PCA is as follows:

$$X = (x_{ij})_{n \times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}. \quad (6)$$

Step 1. Compute the matrix  $V$  which is the covariance matrix of  $X$ .

Step 2. Find out the eigenvalue of  $|V - \lambda E| = 0$ ,  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ .

Step 3. Compute the standardization feature vector of  $(V - \lambda E)\beta = 0$   $\beta_1, \beta_2, \dots, \beta_p$ .

Step 4. Yield the principal components  $Y_r = \beta_r' X$  ( $r = 1, 2, \dots, p$ ).

$E$  is an identity matrix, which has the same dimension with  $V$ . The matrix  $Y$  consists of  $n$  row vectors, where each vector is the projection of the corresponding data vector from matrix  $X$ .

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent if the dataset is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables.

**2.3. Original AdaBoost and Multiclass AdaBoost.** AdaBoost has been very successfully applied in binary classification problem. Original AdaBoost is proposed in [7]. Before proposing the AdaBoost algorithm, the function  $I(x)$  is predefined as

$$I(x) = \begin{cases} 1, & \text{if } x = \text{true} \\ 0, & \text{if } x = \text{false}. \end{cases} \quad (7)$$

AdaBoost algorithm is summarized as follows.

Given the training data  $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $x_i \in R^d$  denotes the  $i$ th input feature vector with  $d$  dimensions,  $y_i$  denotes the label of the  $i$ th input feature vector, where  $y_i \in \{-1, +1\}$ . Use  $T_j(x)$  to denote the  $j$ th weak classifier and suppose  $M$  weak classifiers will be combined.

(1) Initialize the observation weights  $\omega_i = 1/N$ ,  $i = 1, 2, \dots, N$ .

(2) For  $m = 1 : M$ ,

(a) fit a classifier  $T_m(x)$  to the training data using weights  $\omega_i$ ;

(b) compute the weighted error

$$\text{err}_m = \frac{\sum_{i=1}^N \omega_i I(y_i \neq T_m(x_i))}{\sum_{i=1}^N \omega_i}; \quad (8)$$

(c) compute the weight of the  $m$ th classifier

$$\alpha_m = \log \frac{1 - \text{err}_m}{\text{err}_m}; \quad (9)$$

(d) update the weights of sample data, for all  $i = 1, 2, \dots, N$

$$\omega_i = \omega_i \cdot \exp(\alpha_m \cdot I(y_i \neq T_m(x_i))); \quad (10)$$

(e) renormalize  $\omega_i$ , for all  $i = 1, 2, \dots, N$ .

(3) Output

$$C(x) = \arg \max_k \sum_{m=1}^M \alpha_m \cdot I(T_m = k). \quad (11)$$

Here,  $k$  is  $+1$  or  $-1$ . In binary classification, any classifier whose generalization performance is better than  $1/2$  is a weak classifier. For the original AdaBoost, we have the following.

- (1) For the  $i$ th and the  $j$ th classifiers, if  $\text{err}_i < \text{err}_j < 1/2$ , we have  $\alpha_i > \alpha_j > 0$ , which means the final ensemble classifier values more of the  $i$ th classifier's result. Specifically, if  $\text{err}_j = 1/2$ ,  $\alpha_j = 0$ , which means the final ensemble classifier just ignores the classifier since its effect is the same as random guess.
- (2) If the  $p$ th classifier misclassifies the  $q$ th sample, the  $q$ th sample will have a big weight in the next iteration. As a result, the  $(p + 1)$ th classifier will pay more attention to it. On the contrary, if the  $p$ th classifier classifies the  $q$ th sample correctly, the  $q$ th sample will have a small weight in the next iteration, which means  $(p + 1)$ th classifier will pay less attention to it.

However, for a  $K$ -class classification problem, we have  $y_i \in \{1, 2, \dots, K\}$  and  $K > 2$ . If a classifier's generalization performance is better than  $1/K$  (maybe much smaller than  $1/2$ ), it can be called a weak classifier. Since original AdaBoost only takes a classifier whose generalization performance is better than  $1/2$  as a weak classifier, obviously, it cannot be directly implemented to multiclass conditions that  $K$  is bigger than 2. Freund and Schapire [11] extend the original AdaBoost to multiclass condition. The weight of the  $m$ th classifier is modified as

$$\alpha_m = \log \frac{1 - \text{err}_m}{\text{err}_m} + \log(K - 1). \quad (12)$$

Similar to the binary condition, for the  $i$ th and the  $j$ th classifiers, if  $\text{err}_i < \text{err}_j < 1 - 1/K$ , we have  $\alpha_i > \alpha_j > 0$ , which means the final ensemble classifier values more of the  $i$ th classifier's result. In particular, if  $\text{err}_j = 1 - 1/K$ ,  $\alpha_j = 0$ .

**2.4. LBP Based Face Recognition.** The original LBP operator goes through each  $3 \times 3$  neighborhood in a picture. It takes the center pixel as the threshold value of the neighborhood and considers the result as a decimal number. The LBP operator is shown in Figure 1. Then, the texture of the picture can be represented by the histogram of all the decimal numbers.

To apply LBP operator in face recognition problem, Ahonen et al. [12] divided the face image into several windows and calculated the histogram of each window by LBP operator. The final feature vector is gotten by combining the histograms

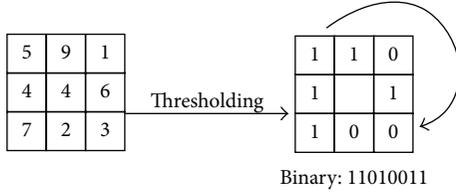


FIGURE 1: Basic LBP operator.

into a spatially enhanced histogram. The spatial enhanced histogram is provided with three levels of information: the patterns of pixel level; the patterns of regional level; the global patterns of the face image. Experiments in [12] have shown that the LBP description is more robust against variants in pose or illumination than holistic methods. All our experiments in Section 4 are done with the most original LBP operator.

### 3. MAELM and Face Recognition Structure

In this part, the multiclass AdaBoost ELM (MAELM) algorithm is proposed and a structure of face recognition based on LBP and ELM is also included.

**3.1. Proposed MAELM Algorithm.** By applying the multiclass AdaBoost to ELM, this paper proposes the multiclass AdaBoost ELM (MAELM) algorithm. The algorithm takes a number of ELM classifiers as the weak classifiers.  $ELM_i(x)$  denotes the  $i$ th ELM classifier. The proposed algorithm is put as follows.

- (1) Initialize the observation weights  $\omega_i = 1/N$ ,  $i = 1, 2, \dots, N$ .
- (2) For  $m = 1 : M$ ,
  - (a) fit a classifier  $ELM_m(x)$  to the training data using weights  $\omega_i$ ;
  - (b) compute the weighted error

$$\text{err}_m = \frac{\sum_{i=1}^N \omega_i I(c_i \neq ELM_m(x_i))}{\sum_{i=1}^N \omega_i}; \quad (13)$$

- (c) compute the weight of the  $m$ th classifier

$$\alpha_m = \log \frac{1 - \text{err}_m}{\text{err}_m} + \log(K - 1); \quad (14)$$

- (d) update the weight of sample data, for all  $i = 1, 2, \dots, N$

$$\omega_i = \omega_i \cdot \exp(\alpha_m \cdot I(c_i \neq ELM_m(x_i))); \quad (15)$$

- (e) renormalize  $\omega_i$ .

- (3) Output

$$C(x) = \arg \max_k \sum_{m=1}^M \alpha_m \cdot I(ELM_m(x) = k). \quad (16)$$

Part (2)(a) of the proposed algorithm should be paid more attention. Both [8, 9] did not give any detail of how to fit the basic classifier  $ELM_m(x)$  with weighted samples, but it is a very important part of AdaBoost. Zong et al. [18] proposed an algorithm named weighted ELM by introducing a diagonal matrix  $W \in R^{N \times N}$ , whose element  $W_{i,i}$  denotes the weight of the  $i$ th training sample. In view of some special situations, we introduce the weighted ELM algorithm. Obviously, it boils down to the original one when the weighted matrix is the identity matrix.

The proposed method maintains the advantages from original ELM: (1) it is simple in theory and convenient in implementation; (2) wide types of feature mapping functions or kernels are available for the proposed framework; (3) the proposed method can be applied directly into multiclass classification tasks. In addition, after integrating with the weighting scheme, the weighted ELM is able to deal with data with imbalanced class distribution while maintaining the good performance on well-balanced data as unweighted ELM; by assigning different weights for each example according to the users' needs, the weighted ELM can be generalized to cost sensitive learning.

Under the weighted circumstance, the solution of  $\beta$  becomes

$$\beta = H^T \left( \frac{I}{C} + WHH^T \right)^{-1} WT. \quad (17)$$

**3.2. Application in LBP Based Face Recognition.** This paper combines LBP based feature vectors with ELM to build a face recognition structure. There have been some papers [19, 20] about applying ELM in face recognition problem. However, the existed ELM based face recognition structures are all based on statistical features, for example, PCA [21] and LDA [22].

In order to get better generalization performance, the proposed face recognition structure implements the LBP based method to get the feature vector and ELM as the classifier. It has been proved in [12] that LBP based method is more robust than PCA and LDA when lighting, facial expression, and poses change. At the same time, ELM is very fast in classification and has very good generalization performance. So, it is reasonable to combine LBP method and ELM to build the face recognition structure.

There are two steps of the proposed face recognition structure. The first step is to train the training samples by ELM or MAELM. In this step, the training samples are represented by LBP based feature vectors. Then, the feature vectors are used to train the classifier model by ELM or MAELM; see Figure 2. The second step is to predict the labels of the test samples. The test samples are also represented by the LBP based feature vectors. Then, the classifier model trained in the first step is implemented to predict the labels of the test samples; see Figure 3.

## 4. Experiments

In this paper, two of the mostly used face recognition datasets Yale and ORL are used to prove the efficiency of the proposed



FIGURE 2: Training the samples by ELM or MAELM.

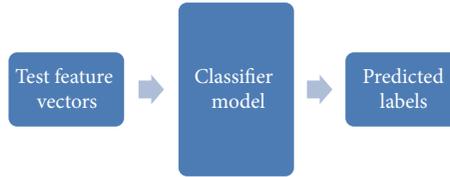


FIGURE 3: Predicting the labels of test samples.

TABLE 1: Parameter list.

| Parameters | Meaning                                      |
|------------|----------------------------------------------|
| $M$        | Number of the basic classifiers              |
| $C$        | Constant value in generalized inverse of $H$ |
| $L$        | Number of hidden nodes in ELM                |
| $t$        | Number of training images of each person     |
| $w$        | Divide each face image into $w * w$ windows  |
| $r$        | The dimension after reduction                |

algorithm. To make the results valid, except for Section 4.2, the average testing accuracy is obtained on 20 trials randomly generated training set and test set. This paper chooses the sigmoid function as the activation function for it is the most commonly used one.

The parameters to set and their meanings in the experiments are listed in Table 1. For example, if the experiment sets  $M = 10$ ,  $C = 1$ ,  $L = 1000$ ,  $t = 5$ , and  $w = 5$ , it means that selecting 5 images of each person builds the training set and the remaining images build the test set. Each image is divided into  $5 \times 5$  windows. After building the training and test set, ELM with  $C = 1$ ,  $L = 1000$  and MAELM, which combines 10 ELMs with  $C = 1$ ,  $L = 1000$ , are evaluated in the built sets.

**4.1. Performance Changes with  $C$  and  $L$ .** Although ELM is comparatively not that sensitive to the arguments as SVM, its performance still changes with the hidden layer number  $L$  and the constant value  $C$ .

Suppose we have  $N$  training samples; Huang et al. [1] rigorously prove that SLFNs (with  $N$  hidden nodes) with random bias and input weights can exactly learn the  $N$  distinct observations. If the training error is allowed, the number of hidden nodes can be much smaller than  $N$ . At the same time, the constant value  $C$  also has some impacts of the solution of  $H$ 's Moore-Penrose generalized inverse.

In this part, the experiment is conducted in Yale dataset. The experiment sets  $M = 20$ ,  $t = 5$ , and  $w = 3$ . In addition,

the  $L$  is set as 100, 400, 700, ..., 1900 and the  $C$  is set as  $10^{-5}$ ,  $10^{-4}$ , ..., 1,  $10^1$ ,  $10^2$ , ...,  $10^5$ . The performance of ELM and MAELM is shown in Figure 4.

It is obvious that both ELM and MAELM are not sensitive to the change of arguments. The difference between ELM and MAELM is mainly in the region where  $L$  is very small and  $C$  is very large. From Figure 4, one can conclude that ELM performs badly in this region, since its accuracy rate is below 0.6. On the contrary, MAELM is still very stable in this region. Its accuracy rate is bigger than 0.8.

After seeing PCA's good performance in the region of face recognition, we wonder if PCA could have a stable and better performance when it replaces the way we originally construct the matrix  $H$ .

The experiment is also conducted in Yale dataset with the same parameters. Besides, the new parameter  $r$ , which is the dimension after reduction, could not be set bigger than the number of input nodes. In view of the dimension of dataset and other limitations in the experiment, the parameter  $r$  is set as 10, 20, ..., 60. Since it is complex in the picture because of the imbalance with the parameters change, we choose to show them in the table. The performance of ELM (Figure 4(a)) and MAELM (Figure 4(b)) with PCA is listed in Table 2; the best accuracy rate in the table is bold.

It is clear that both ELM and MAELM with PCA are not so sensitive to the change of arguments. The difference between them is mainly in the region where  $L$  is very small and  $C$  is very large. From Table 2, one can conclude that MAELM with PCA performs better in this region when  $C$  is very small, but when  $C$  is large and  $r$  is small, ELM with PCA performs rather well and stable. Besides, ELM with PCA's performance is almost as well as the other one in the region where  $C$  and  $r$  are both very large, and its accuracy rate is bigger than 0.85.

**4.2. Prediction Stability Analysis.** Since the original ELM randomly generates the weights between the input layer and the hidden layer, as well as the bias of the activation function, its performance even for the same training and test set changes each time. This is to say the performance of

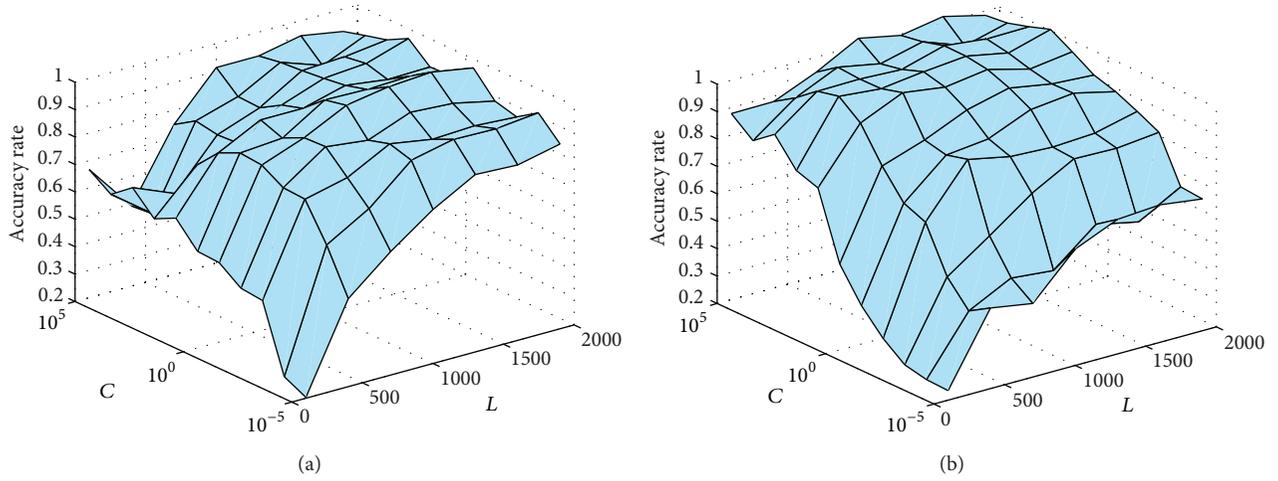


FIGURE 4: The performance of ELM (a). The performance of MAELM (b).

TABLE 2: Performance of ELM and MAELM with PCA.

| $C/r$     | 10                | 20                | 30                | 40                | 50                | 60                |
|-----------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| $10^{-5}$ | 0.19/ <b>0.38</b> | 0.29/ <b>0.33</b> | 0.2/ <b>0.32</b>  | 0.23/ <b>0.41</b> | 0.09/ <b>0.15</b> | 0.22/ <b>0.30</b> |
| $10^{-4}$ | 0.19/ <b>0.20</b> | 0.22/ <b>0.40</b> | <b>0.21/0.21</b>  | <b>0.36/0.35</b>  | 0.27/ <b>0.28</b> | 0.26/ <b>0.38</b> |
| $10^{-3}$ | 0.27/ <b>0.39</b> | <b>0.38/0.35</b>  | 0.24/ <b>0.46</b> | <b>0.37/0.36</b>  | <b>0.33/0.31</b>  | <b>0.38/0.30</b>  |
| $10^{-2}$ | <b>0.43/0.34</b>  | <b>0.76/0.32</b>  | <b>0.85/0.32</b>  | <b>0.86/0.43</b>  | <b>0.86/0.37</b>  | <b>0.86/0.35</b>  |
| $10^{-1}$ | <b>0.79/0.36</b>  | <b>0.84/0.43</b>  | <b>0.88/0.40</b>  | <b>0.88/0.36</b>  | <b>0.90/0.42</b>  | <b>0.91/0.53</b>  |
| $10^0$    | <b>0.84/0.58</b>  | <b>0.95/0.76</b>  | <b>0.86/0.81</b>  | <b>0.90/0.85</b>  | <b>0.87/0.82</b>  | <b>0.91/0.87</b>  |
| $10^1$    | <b>0.77/0.66</b>  | <b>0.90/0.87</b>  | <b>0.89/0.88</b>  | <b>0.91/0.91</b>  | <b>0.98/0.97</b>  | <b>0.91/0.88</b>  |
| $10^2$    | <b>0.77/0.73</b>  | 0.85/ <b>0.87</b> | <b>0.91/0.91</b>  | 0.90/ <b>0.92</b> | <b>0.94/0.92</b>  | <b>0.92/0.92</b>  |
| $10^3$    | <b>0.77/0.74</b>  | <b>0.90/0.89</b>  | 0.92/ <b>0.93</b> | <b>0.94/0.94</b>  | 0.93/ <b>0.95</b> | 0.90/ <b>0.92</b> |
| $10^4$    | <b>0.74/0.55</b>  | <b>0.91/0.88</b>  | <b>0.93/0.93</b>  | <b>0.88/0.88</b>  | <b>0.91/0.91</b>  | 0.85/ <b>0.86</b> |
| $10^5$    | <b>0.79/0.71</b>  | <b>0.87/0.87</b>  | <b>0.87/0.87</b>  | <b>0.97/0.97</b>  | <b>0.96/0.96</b>  | <b>0.91/0.91</b>  |

original ELM may not be so stable. The proposed algorithm successfully reduces the instability.

From Figure 4, one is able to conclude that ELM tends to get better performance when  $C = 1$ , while  $C = 10^3$  is better for MAELM. Let  $M = 10$ ,  $C = 10^3$ ,  $L = 1000$ ,  $t = 5$ , and  $w = 3$  for MAELM and  $C = 1$ ,  $L = 1000$ ,  $t = 5$ , and  $w = 3$  for ELM. Besides, ELM and MAELM ( $r = 20$ ) with PCA are also included under the corresponding situations because of the considerate performance above. Experiments are done in Yale datasets. In order to prove that the proposed algorithm is more stable than the original ELM, experiments are done in the same training set and test set (randomly generated) for 20 times. The result is shown in Figure 5.

In Figure 5, it is obvious that the performance of MAELM is much more stable than the original ELM. Although ELM or MAELM with PCA performs far more stable than the original ELM and MAELM (since they take the algorithm of PCA into consideration instead of the random weights between the input layer and the hidden layer and the bias of the activation function), the accuracy rates of them, which are always in the middle from Figure 5, are still not so good as the original MAELM. We conclude the result of Figure 5 in Table 3. Please notice that although the generalization performance

TABLE 3: Performance of ELM and MAELM under the same training set and test set.

| Algorithm | Mean accuracy rate | Standard derivation |
|-----------|--------------------|---------------------|
| ELM       | 0.8972             | 0.0213              |
| MAELM     | 0.9361             | 0.0157              |
| ELM.PCA   | 0.9222             | 0                   |
| MAELM.PCA | 0.9222             | 0                   |

of MAELM seems to be much better than ELM in the table, it is improper to conclude that MAELM performs better. The reason is that the training set and test set are fixed. One cannot exclude the possibility that MAELM performs better than ELM only under this dataset. Some other experiments will be done in the following parts to show MAELM's better generalization performance.

4.3. *Performance Changes with  $M$ .* In order to evaluate the changes of performance when  $M$  changes, the experiment in this part lets  $C = 1$ ,  $t = 5$ ,  $w = 4$ ,  $L = 1000$  for the original MAELM,  $r = 20$  for MAELM with PCA, and  $M = 2, 4, 6, \dots, 50$ . The average test accuracy is obtained on

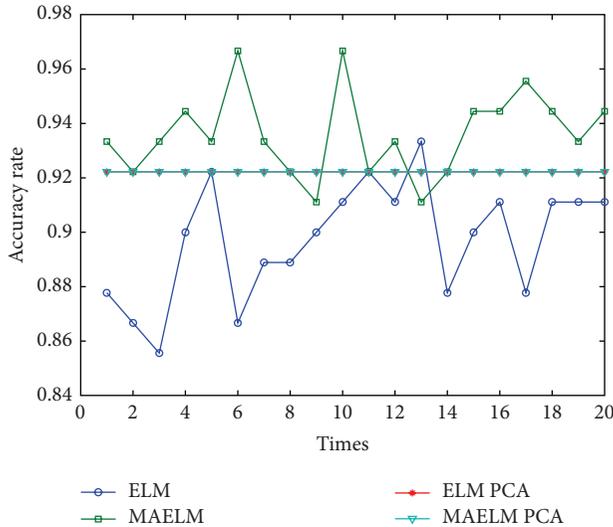


FIGURE 5: Performance of ELM and MAELM under the same training set and test set.

20 trials randomly generated training set and test set. Yale dataset is used for the experiment. The result is presented in Figures 6 and 7.

From Figure 6, it is obvious that as the  $M$  increases, the generalization performance also becomes better. However, the trend becomes slower as  $M$  increases. From Figure 7, one can conclude that as the  $M$  increases when  $M$  is small, the performance decreases a little, while  $M$  becomes larger after 25; the performance also becomes better, although the trend is not so stable as the original MAELM. This situation indicates that in real-world applications,  $M$  does not need to be very big. Good generalization performance can be obtained by setting  $M$  less than 30 in the algorithm of original MAELM, which achieves better than MAELM with PCA under the same situation.

**4.4. Better Generalization Performance Than ELM.** In this part, experiments are done both in Yale and ORL datasets. The experiments set the parameters of those algorithms as follows:  $C = 1$ ,  $L = 1000$ ,  $t = 5$ ,  $M = 20$  (MAELM), and  $r = 20$  (PCA). The experiments take  $3 \times 3$ ,  $4 \times 4$ ,  $5 \times 5$ ,  $6 \times 6$ , and  $7 \times 7$  windows into consideration, which means setting  $w = 3, \dots, 7$ . The average testing accuracy is obtained on 20 trials randomly generated training set and test set.

The experiment indicates that MAELM has better generalization performance both in Yale and ORL datasets under different window sizes. See Figure 8 for details, while in Figure 9, it is obvious that ELM with PCA has much better performance both in Yale and ORL datasets under different window sizes. In addition this algorithm keeps more stable than any other algorithms both in Yale and ORL datasets.

**4.5. The Performance in PCA.** After seeing all these experiments, we can conclude that although MAELM with PCA performs not so well as the original one, ELM with PCA

performs much better than before, especially in the experiment in Section 4.2. It is obvious that the performance of the experiments with PCA is just between the original ELM and MAELM.

What is more, since the original ELM randomly generates the weights between the input layer and the hidden layer, as well as the bias of the activation function, its performance is not so stable. The proposed algorithm with PCA successfully reduces the instability which is very important in the real world.

Although PCA improves the performance of ELM in a certain degree, it still could not reach the ability of MAELM with random weights and bias. Finally, it comes to the result that the proposed algorithm named MAELM performs much better in solving the multiclass classification problem.

## 5. Discussion

**5.1. Complexity Comparison.** Very similar to MAELM, the DAEELM [8] also considers taking the ELM as the weak classifier and implements AdaBoost as the ensemble method. The difference is that MAELM implements multiclass AdaBoost which can be directly used in multiclass classification problem, while DAEELM implements dynamic ensemble AdaBoost [23], which aims to solve the binary classification problem.

Many methods have been developed to apply binary classifier to multilabel problem. One-against-all (OAA) [24] and one-against-one (OAO) [25] are mostly used. For a  $K$ -class classification problem, under OAA condition,  $K$  classifiers have to be trained. Each of them separates a single class from all the remaining classes. Under the OAO condition,  $K(K-1)/2$  classifiers have to be trained. Each of them separates a pair of classes.

Suppose that both MAELM and DAEELM have  $M$  iterations. For a  $K$ -class classification problem, MAELM only needs to train  $M$  ELMs, while DAEELM needs to train  $M \times K$  and  $(M \times K \times (K-1))/2$  classifiers for OAA and OAO condition, respectively. Although DAEELM may stop the iteration earlier, it is obvious that, in theory, MAELM's computation complexity is much lower than DAEELM for  $K$ -class classification problem, especially when  $K$  is a very big number.

The authors of DAEELM have not published its codes and DAEELM has its own arguments which MAELM does not have. DAEELM also does not provide details of how it trains weighted data with ELM, so it will be unfair to compare the performance of MAELM and DAEELM. However, the conclusion that MAELM is much faster than DAEELM in multiclass classification problem can be drawn from the complexity analysis above.

**5.2. Train ELM with Weighted Data.** Section 3.1 has mentioned that training ELM with weighted data is a key problem when applying AdaBoost. However, [8, 9] did not mention the key point at all.

Toh in [26] first applied ELM to classify imbalanced data with two classes. ELM tries to minimize the training error of

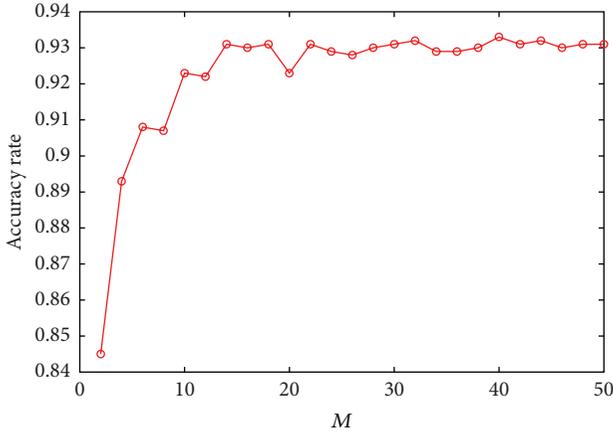


FIGURE 6: MAELM's performance.

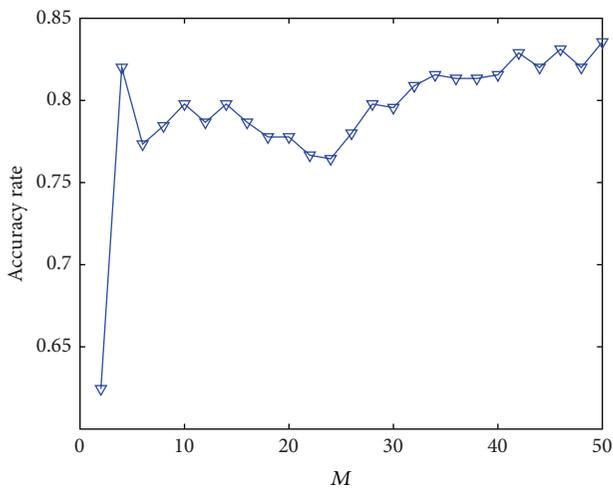


FIGURE 7: MAELM with PCA's performance.

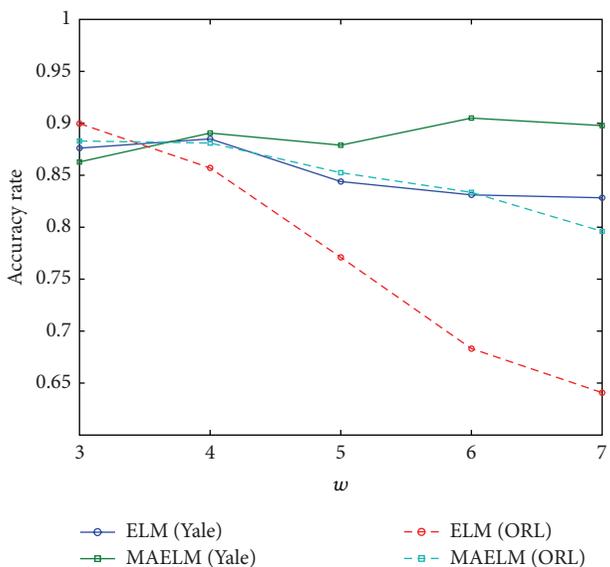


FIGURE 8: Performances in Yale and ORL.

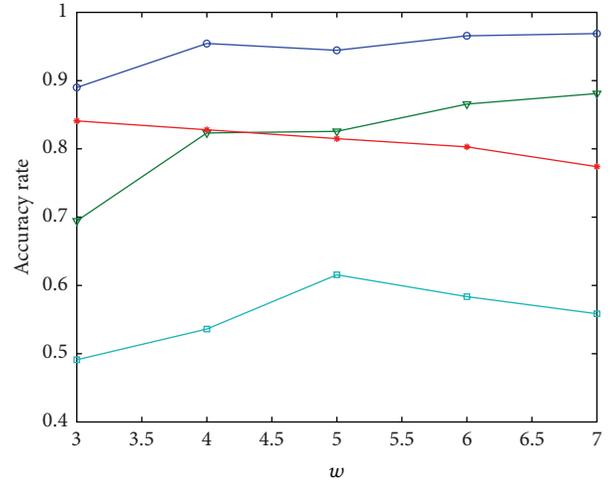


FIGURE 9: Performances in Yale and ORL.

the data while the proposed algorithm tends to minimize the total error rate (TER), which takes the weights of the positive and negative data into consideration.

In Section 3.1, the weighted ELM is applied in MAELM. Actually, the weighted ELM is inspired and in a way that is very similar to regularized ELM proposed by Deng et al. in [27]. The regularized ELM aims to minimize the weighted training error of the weighted data.

## 6. Conclusion

This paper proposes a new boosting ELM named MAELM, which applies the multiclass AdaBoost in ELM ensemble to directly solve multiclass classification problem. A face recognition structure combined LBP based method and ELM is also presented in the paper. What is more, this paper proposes the way in which ELM combined with PCA instead of using random weights between the input layer and the hidden layer, as well as the bias of the activation function.

Experiments in LBP based face recognition will show the stable and good performance in a certain degree. Although PCA improves the performance of ELM, it still could not be better than MAELM with random weights and bias. Experiments show that in LBP based face recognition problem, the recognition result of MAELM is more stable than the original ELM and better than any other algorithms listed in the paper.

Finally, it comes to the result that the proposed algorithm named MAELM, which applies the multiclass AdaBoost in ELM and combines with LBP method, performs much better in solving the multiclass classification problem.

Also, MAELM is compared with DAEELM in multi-class classification problem in theory, which indicates that MAELM has much lower computation complexity than DAEELM. Moreover, this paper makes the problem how to train weighted data by ELM clear.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This research is based on work supported in part by the National Natural Science Foundation of China (61370173, 61173123) and the Natural Science Foundation Project of Zhejiang Province under Project LR13F030003.

## References

- [1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.
- [2] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376-390, 2014.
- [3] P. K. Wong, Z. Yang, C. M. Vong, and J. Zhong, "Real-time fault diagnosis for gas turbine generator systems using extreme learning machine," *Neurocomputing*, vol. 128, pp. 249-257, 2014.
- [4] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronic and Application*, pp. 1163-1168, 2014.
- [5] J. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [6] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66-77, 2012.
- [7] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337-407, 2000.
- [8] G. Wang and P. Li, "Dynamic Adaboost ensemble extreme learning machine," in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, pp. V3-54-V3-58, IEEE, Chengdu, China, August 2010.
- [9] H.-X. Tian and Z.-Z. Mao, "An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 73-80, 2010.
- [10] D. P. Solomatine and D. L. Shrestha, "AdaBoost.RT: a boosting algorithm for regression problems," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 1163-1168, 2004.
- [11] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.
- [12] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Computer Vision-ECCV 2004*, vol. 3021 of *Lecture Notes in Computer Science*, pp. 469-481, Springer, Berlin, Germany, 2004.
- [13] Y. Shen, Y. L. Jiang, W. Liu, and Y. Liu, "Multi-class AdaBoost ELM," in *Proceedings of the International Conference on Extreme Learning Machines (ELM '14)*, Singapore, December 2014.
- [14] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513-529, 2012.
- [15] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 11, no. 2, pp. 559-572, 1901.
- [16] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 7, pp. 498-520, 1933.
- [17] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433-459, 2010.
- [18] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229-242, 2013.
- [19] W. Zong and G.-B. Huang, "Face recognition based on extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2541-2551, 2011.
- [20] A. A. Mohammed, R. Minhas, Q. M. Jonathan Wu, and M. A. Sid-Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2588-2597, 2011.
- [21] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.
- [22] K. Etamad and R. Chellappa, "Discriminant analysis for recognition of human face images," *Journal of the Optical Society of America A*, vol. 14, no. 8, pp. 1724-1733, 1997.
- [23] R. Li, J. Lu, Y. Zhang, and T. Zhao, "Dynamic Adaboost learning with feature selection based on parallel genetic algorithm for image annotation," *Knowledge-Based Systems*, vol. 23, no. 3, pp. 195-201, 2010.
- [24] B. Heisele, P. Ho, J. Wu, and T. Poggio, "Face recognition: component-based versus global approaches," *Computer Vision and Image Understanding*, vol. 91, no. 1-2, pp. 6-21, 2003.
- [25] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *The Journal of Machine Learning Research*, vol. 1, no. 2, pp. 113-141, 2001.
- [26] K.-A. Toh, "Deterministic neural classification," *Neural Computation*, vol. 20, no. 6, pp. 1565-1595, 2008.
- [27] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pp. 389-395, Nashville, Tenn, USA, April 2009.

## Research Article

# Daily Human Physical Activity Recognition Based on Kernel Discriminant Analysis and Extreme Learning Machine

Wendong Xiao and Yingjie Lu

*School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China*

Correspondence should be addressed to Wendong Xiao; [wdxiao@ustb.edu.cn](mailto:wdxiao@ustb.edu.cn)

Received 10 September 2014; Accepted 26 November 2014

Academic Editor: Amaury Lendasse

Copyright © 2015 W. Xiao and Y. Lu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Wearable sensor based human physical activity recognition has extensive applications in many fields such as physical training and health care. This paper will be focused on the development of highly efficient approach for daily human activity recognition by a triaxial accelerometer. In the proposed approach, a number of features, including the tilt angle, the signal magnitude area (SMA), and the wavelet energy, are extracted from the raw measurement signal via the time domain, the frequency domain, and the time-frequency domain analysis. A nonlinear kernel discriminant analysis (KDA) scheme is introduced to enhance the discrimination between different activities. Extreme learning machine (ELM) is proposed as a novel activity recognition algorithm. Experimental results show that the proposed KDA based ELM classifier can achieve superior recognition performance with higher accuracy and faster learning speed than the back-propagation (BP) and the support vector machine (SVM) algorithms.

## 1. Introduction

Recognition of physical activity plays an important role in many fields such as physical training and health care. In particular nowadays, faced with the aging problem, a growing number of old people live alone and urgently demand advanced solutions for their health monitoring, including their physical activity recognition.

A number of key research issues are related to physical activity recognition, including how to improve the data collection mechanisms, how to select more effective features, and how to design high-performance classification algorithms. Different solutions have been proposed to address these issues, usually based on the video analysis and the wearable sensor signal analysis. The video analysis approach needs to obtain the position and attitude information from a series of body image sequences. Due to the complexity and the variability of the human physical activities, it often suffers from low accuracy and low efficiency in many practical scenarios. It is also weak for privacy of the monitored person [1, 2]. With the development of microelectromechanical system (MEMS) and wearable sensor networks, activity recognition approaches based on the wearable sensors, especially

accelerometers, have received increasing interests, due to their advantages that can be implemented easily anywhere anytime [3–8].

In previous studies, many researchers use multiple sensors for physical activity recognition to improve the accuracy. But the hardware would obstruct the movements of the human and is not practical for long-term wearing. Also the cost of system will increase with the number of sensors [3–5]. Therefore recently more researchers are seeking activity recognition approaches by using only one accelerometer sensor for collecting the signal [6–8].

Feature extraction is crucial for activity recognition. It is required to extract useful features from the raw measurement data to reduce the processing time and improve the recognition accuracy. Based on the accelerometer data, a number of features and feature extraction methods have been proposed, covering from the time domain analysis and the frequency domain analysis to the time-frequency domain analysis [4, 9]. The time domain method extracts features, such as the mean, the standard deviation, and the correlation coefficient, from the collected signal directly. The frequency domain method extracts features from the frequency-domain parameters, such as FFT coefficients. More recently, wavelet analysis,

which can incorporate time and frequency information, has been used to perform the time-frequency analysis.

Classifier is the key for activity recognition. Besides the high accuracy and short training time, the classifier is often expected to meet the real-time and the generalization requirements. A large number of classification methods have been investigated, including the artificial neural network (ANN) [7] and the support vector machine (SVM) [10], which have been widely used in machine learning and data analysis. But these popular learning techniques often face some challenging issues such as intensive human intervene, slow learning speed, and poor learning scalability [11–13]. Therefore, the recent extreme learning machine (ELM) [11] classifier will be proposed in this paper as a highly efficient and accurate activity recognition approach.

ELM is developed as a single-hidden layer feed-forward network (SLFN) that can randomly assign the weights between the input nodes and the hidden nodes and analytically determine the output weights between the hidden nodes and output nodes. It learns much faster than traditional gradient-based approaches, such as back-propagation (BP) algorithm. ELM tends to reach the small norm of the network output weights and achieve better generalization performance according to Bartlett's theory states [12]. In theory, ELM with the same kernels outperforms SVM in both regression and classification applications [13, 14]. ELM provides efficient unified solutions to generalized feed-forward networks including but not limited to neural networks, radial basis function (RBF) networks, and kernel learning [15]. Many improvements of ELM are under study in recent years, such as fast online learning [16], large scale ELM [17], and voting based ELM [18]. Also various applications can be found based on ELM and its variants, including wireless indoor localization [19], multicollinear problem [20, 21], and protein sequence classification [22].

The patterns of the physical activities vary from simple activities (such as standing, sitting, and walking) to more complex action strings (such as eating, drinking, and cycling). With the addition of new activities, the features of the existing systems may be ineffective, and the recognition accuracy may decrease significantly. To find new features will increase the workload of the researchers, and classifier design will become difficult with the increase of the number of the features. Principal component analysis (PCA) [23] and linear discriminant analysis (LDA) [7, 24] are applied to select most discriminative features for activity recognition. Kernel discriminant analysis (KDA) [10, 25] is an extension of LDA to obtain nonlinear discriminating features by the kernel technique for mapping the data to the feature space. In this paper, we will introduce KDA to extract more meaningful features of the activities and integrate it with ELM classifier to achieve improved classification performance.

The paper is organized as follows. The proposed approach is detailed in Section 2, including its general description, system design, data acquisition and preprocessing, feature extraction method, KDA, and the ELM algorithm. Experimental results are reported in Section 3. Finally, conclusions and the future work are given in Section 4.

## 2. Proposed Approach

The proposed activity recognition approach uses a triaxial accelerometer for data collection. As shown in Figure 1, the overall approach consists of the following steps: data acquisition, preprocessing, feature extraction, KDA, ELM training and classification.

*2.1. Wearable Component.* The wearable sensor in this paper employs a MPU-6000 sensor, which contains a triaxial accelerometer and a triaxial gyroscope. The triaxial accelerometer is used to collect the raw acceleration measurement signal. The sampling frequency is set at 50 Hz with the output ranging in  $[-4g, +4g]$ . The sensor transmits the data to a mobile phone via Bluetooth module called HC-05 and the data are stored in the SD card of the phone.

The wearable sensor is small and convenient to carry and is worn on the subject's thigh of right leg, as shown in Figure 2. When the body wears the accelerometer at the state of standing, the  $x$ -axis represents the acceleration in the lateral direction, the  $y$ -axis represents the acceleration in the longitudinal direction, and the  $z$ -axis represents the acceleration in the vertical direction. In order to facilitate observations, the measurement of each axis is divided by the gravitational acceleration (taken as  $9.8 \text{ m/s}^2$ ); therefore, the data are multiples of the gravitational acceleration.

*2.2. Data Acquisition and Preprocessing.* In the experiments, 10 subjects' data are collected, 5 females and 5 males, wearing the wearable sensor each day for a period to collect data for six different activities of daily life, including sitting, standing, walking, running, going upstairs, and going downstairs. Figure 3 shows the example acceleration signals of sitting and walking for each axis of the triaxial accelerometer.

As the sensor will be influenced by the gravity, the data are filtered by a high-pass filter with a cut-off frequency of 0.5 Hz to eliminate the influence of the gravity. As the frequency of human daily activities is not too high, a low-pass filter with a cut-off frequency of 20 Hz is used to filter high frequency noise. In addition, the data are smoothed by median filter to eliminate independent noise.

*2.3. Feature Extraction.* Features are extracted from the raw accelerometer data over the sliding window. In this paper, features were computed from 128 sampling points (2.56 s) with 64 samples overlapping between the consecutive sliding windows. Feature extraction on sliding windows with 50% overlap has been shown to be effective in previous studies on activity classification [4].

A number of features, including the mean, the standard deviation, the median, the correlation, the tilt angle (TA), the signal magnitude area (SMA), the frequency energy, the frequency entropy, and the wavelet energy, were extracted from the signals for activity recognition. A brief description of each feature is given as follows.

(1) *Tilt Angle (TA).* No matter what the state of the device is, there will be vertical gravity. When tilting the device, there will be a gravitational component in the  $z$ -axis. The tilt angle

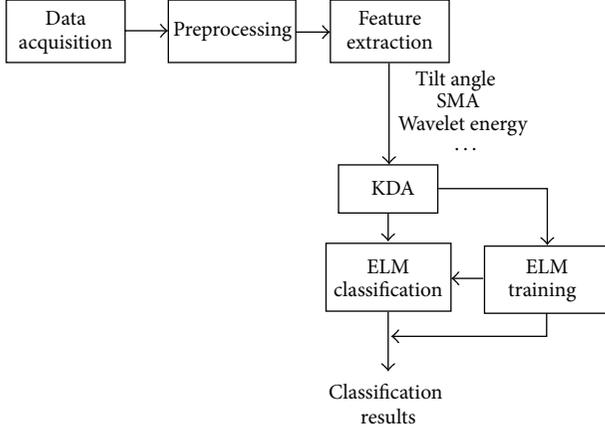


FIGURE 1: The architecture of the proposed activity recognition approach.

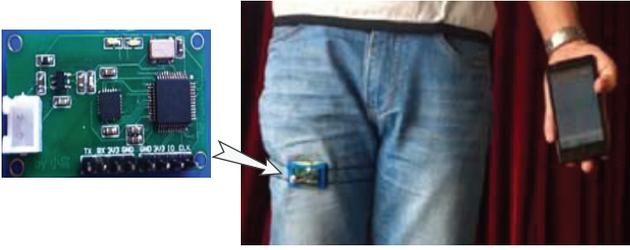


FIGURE 2: Wearable sensor being worn by a subject.

refers to the relative tilt of the body in space and is calculated from the acceleration in the  $z$ -axis according to

$$\theta = \arccos \frac{a_z}{g}, \quad (1)$$

where  $a_z$  is the gravity component in the  $z$ -axis and  $g$  is the gravity coefficient.

(2) *Signal Magnitude Area (SMA)*. We adopt the SMA to extract a feature quantity according to

$$SMA = \sum_{i=1}^N (|x(i)| + |y(i)| + |z(i)|), \quad (2)$$

where  $x(i)$ ,  $y(i)$ , and  $z(i)$  indicate the values of  $x$ -axis,  $y$ -axis, and  $z$ -axis acceleration signals at the  $i$ th sampling point after preprocessing and  $N$  is the length of the sliding window. SMA can indicate the fluctuation degree of the acceleration signal; the higher its value is, the more violent the fluctuation is.

(3) *Wavelet Energy (WE)*. This paper uses db5 as the mother wavelet. By decomposing the vertical component to 5 layers, WE is calculated as the sum of the squared detail coefficients at levels 4 and 5, according to

$$WE = \sum_{j=4}^5 CD_j^2, \quad (3)$$

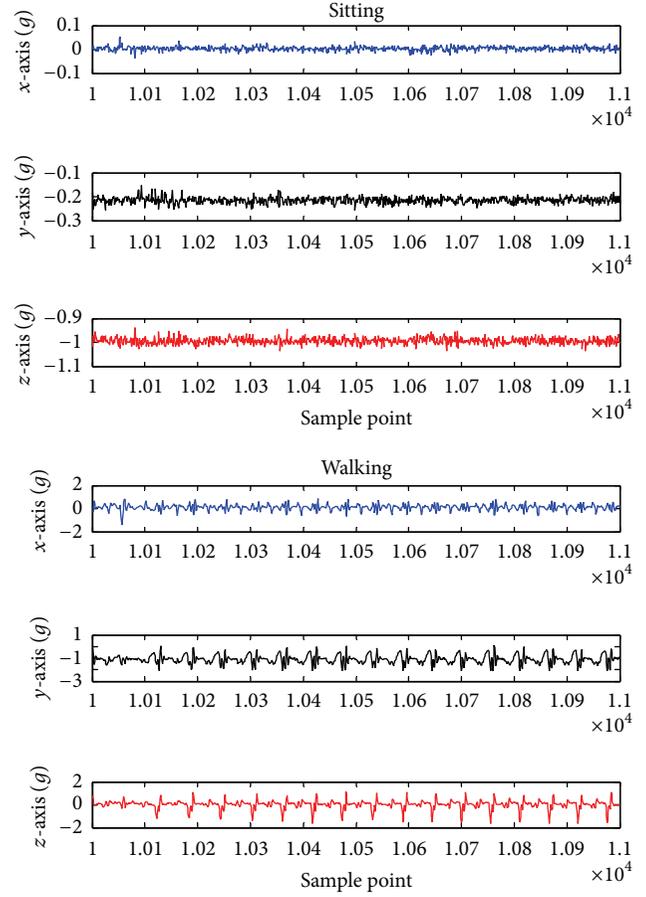


FIGURE 3: The raw acceleration signals of sitting and walking.

where  $CD_j$  are the detail coefficients of  $y$ -axis acceleration signal.

(4) *Other Features*. Mean, standard deviation, and median are the average, standard deviation, and median value of the signal over the sliding window, respectively. Correlation coefficient is calculated between the  $y$ -axis and  $z$ -axis of accelerometer signal. The frequency energy feature is calculated as the sum of the squared magnitudes of the discrete FFT components of the signal. Entropy is calculated as the normalized information entropy magnitudes of the discrete FFT components of the signal.

Figure 4 shows some of these features for different activities. We can find that it is easy to distinguish some activities, such as sitting and standing, but the differences among walking, going upstairs, and going downstairs are not significant, likely to cause misrecognition.

2.4. *Kernel Discriminant Analysis (KDA)*. In order to improve the classification accuracy, further operations on the derived features are performed.

Linear discriminant analysis (LDA) is a supervised dimensionality reduction technique used for data analysis and pattern recognition. LDA tries to maximize the separation between different classes and minimize the separation

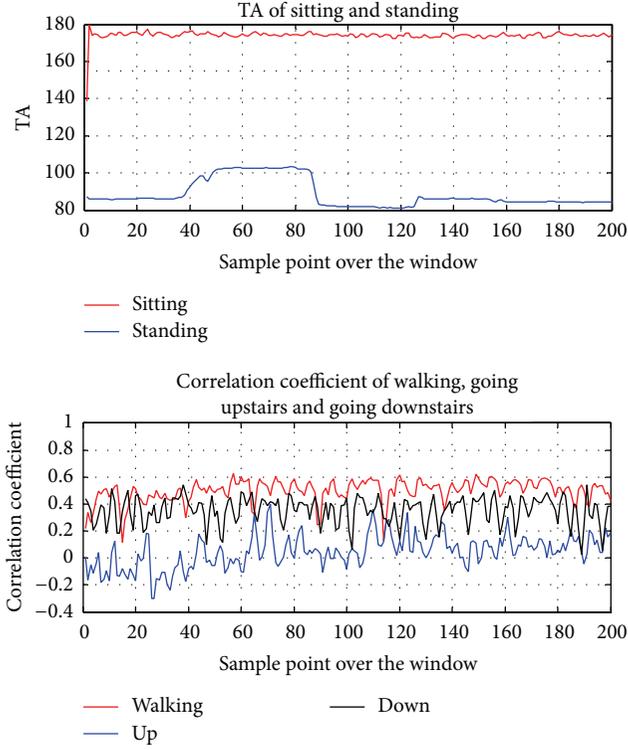


FIGURE 4: Difference in feature values for discriminating different activities.

within the same class simultaneously. KDA is a nonlinear extension of LDA to obtain nonlinear discriminating features by the kernel technique [26]. In KDA, input data are mapped to the high dimensional feature space  $F$  by nonlinear feature mapping  $\phi : R^n \rightarrow F$ . We select Gaussian radial basis function (RBF) for  $\phi$ . In KDA, the cross class and intraclass scatter matrices are computed as

$$S_B^\phi = \sum_{i=1}^c N_i (\mu_i^\phi - \mu^\phi) (\mu_i^\phi - \mu^\phi)^T, \quad (4)$$

$$S_W^\phi = \sum_{i=1}^c \sum_{x \in X_i} (\phi(x) - \mu_i^\phi) (\phi(x) - \mu_i^\phi)^T,$$

where  $N_i$  is the number of samples from the  $i$ th class,  $c$  is the number of classes,  $\mu_i^\phi$  is the centroid of the  $i$ th class and  $\mu^\phi$  is the global centroid,  $x$  is a vector for a specific class, and  $X_i$  is the set of samples of the  $i$ th class.  $S_W^\phi$  represents the degree of scattering within classes of activities and is calculated as the summation of covariance matrices of each class, whereas  $S_B^\phi$  represents the degree of scattering between classes of activities and is calculated as the summation of the covariance matrix of the means of each class. The optimal discrimination transformation in the projection space is obtained by solving the following optimization problem:

$$\omega_{\text{opt}} = \arg \max_{\omega} \frac{\omega^T S_B^\phi \omega}{\omega^T S_W^\phi \omega}. \quad (5)$$

The optimization solution  $\omega_{\text{opt}}$ , corresponding to the largest eigenvalues  $\lambda$ , can be explained by the generalized eigenvalue problem:

$$S_B^\phi \omega_i = \lambda_i S_W^\phi \omega_i. \quad (6)$$

It is proved that the above equation can be equivalently represented as

$$A_{\text{opt}} = \arg \max_A \frac{A^T K W K A}{A^T K K A}. \quad (7)$$

The corresponding eigenvalue problem is represented as

$$K W K A = \lambda K K A, \quad (8)$$

where  $K$  is the kernel matrix with its element defined as

$$K_{ij} = k(x_i, x_j), \quad (9)$$

where  $k(\cdot, \cdot)$  is a positive semidefinite kernel function and  $W$  is a matrix with its element defined as

$$W_{ij} = \begin{cases} \frac{1}{n_c}, & \text{if } x_i \text{ and } x_j \text{ belong to the } c\text{th class} \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

More details on KDA can be found in [26].

**2.5. ELM Algorithm.** Extreme learning machine (ELM) was introduced by Huang et al. [11], originally proposed for standard single-hidden layer feed-forward neural networks (SLFNs), with random hidden nodes, and has recently been extended to kernel learning as well [27]. Compared with traditional training methods, ELM has the advantages of high accuracy, fast learning speed, and good generalization property. It can provide a unified learning platform with widespread type of feature mappings and can be applied in regression and multiclass classification applications directly.

Figure 5 is the structure of ELM. The network consists of an input layer, a hidden layer, and an output layer. For  $N$  arbitrary distinct samples  $(x_i, t_i)$ , denote the input signal vector  $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T \in R^m$  and the output signal vector  $t_i = [t_{i1}, t_{i2}, \dots, t_{in}]^T \in R^n$ , and  $w_j = [w_{j1}, w_{j2}, \dots, w_{jm}]^T$  is the weight vector connecting the  $j$ th hidden node and the input nodes,  $\beta_j = [\beta_{j1}, \beta_{j2}, \dots, \beta_{jn}]^T$  is the weight vector connecting the  $j$ th hidden node and the output nodes, and  $b_j$  is the threshold of the  $j$ th hidden node.

The activation function of hidden layer neurons is  $g(x)$ , and the hidden layer has  $L$  neurons. There exist  $w_j$ ,  $\beta_j$ , and  $b_j$  such that

$$\sum_{j=1}^L \beta_j g(w_j x_i + b_j) = t_i, \quad i = 1, \dots, N. \quad (11)$$

Denote the output of the network as

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times n}. \quad (12)$$

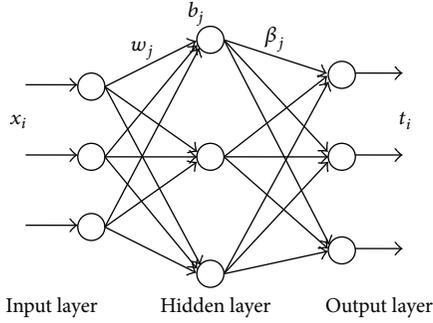


FIGURE 5: The structure of ELM.

Then (11) can be rewritten as

$$H\beta = T, \quad (13)$$

where  $H$  is the hidden layer output matrix of the ELM:

$$H(w_1, w_2, \dots, w_L, b_1, b_2, \dots, b_L, x_1, x_2, \dots, x_N) = \begin{bmatrix} g(w_1x_1 + b_1) & g(w_2x_1 + b_2) & \cdots & g(w_Lx_1 + b_L) \\ g(w_1x_2 + b_1) & g(w_2x_2 + b_2) & \cdots & g(w_Lx_2 + b_L) \\ \vdots & \vdots & \ddots & \vdots \\ g(w_1x_N + b_1) & g(w_2x_N + b_2) & \cdots & g(w_Lx_N + b_L) \end{bmatrix}_{N \times L},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times n}.$$

(14)

According to the theorem proven in [11], when the activation function is infinitely differentiable, and  $L \leq N$ , the parameters of SLFNs do not need all to be adjusted. Parameters  $w$  and  $b$  can be selected randomly before training and remain constant during the training process. To train SLFNs is simply equivalent to finding a least-square solution  $\hat{\beta}$  of the linear system  $H\beta = T$ ; that is,

$$\|H\hat{\beta} - T\| = \min_{\beta} \|H\beta - T\|. \quad (15)$$

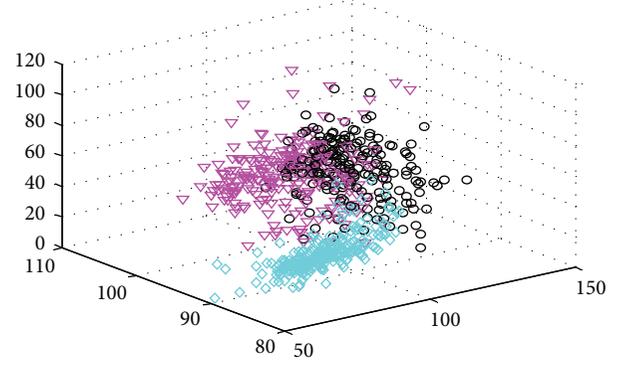
Then the optimal output weights can be calculated as

$$\beta = H^\dagger T, \quad (16)$$

where  $H^\dagger$  is the Moore-Penrose generalized inverse of the matrix  $H$ .

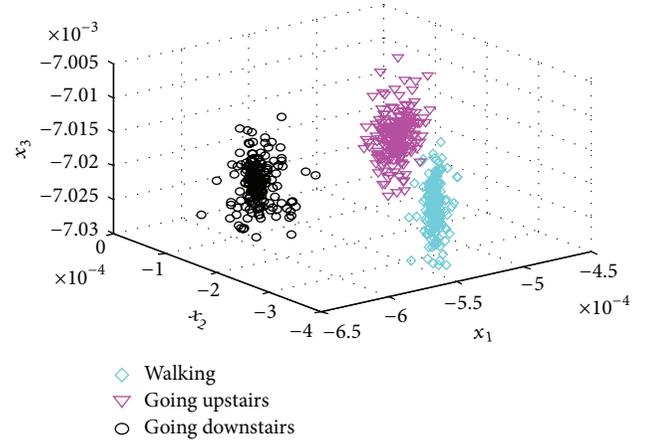
### 3. Approach Implementation and Experimental Results

In this section, we will implement the approach for extracting features of the data, using KDA to map the features to the high dimensional feature space, applying ELM algorithm to classify the samples and comparing ELM with other classic classifiers.



◇ Walking  
▽ Going upstairs  
○ Going downstairs

(a) 3D feature space representation on original features



◇ Walking  
▽ Going upstairs  
○ Going downstairs

(b) 3D feature space representation for KDA implementation on original features

FIGURE 6: Features without and with KDA operations.

**3.1. Recognition Using KDA on Original Features.** The difference of the features among the three activities (walking, going upstairs, and going downstairs) is not so significant. Therefore, KDA is implemented to deal with the problem. The 3D plot in Figure 6 shows the low intraclass variance and high cross class variance for the three activities. Figure 6(a) shows the three features, SMA, TA, and WE, extracted from the collected samples. It is difficult to classify them. Figure 6(b) shows significantly improved discrimination among the three activities after the KDA operation.

**3.2. Classification Results.** In this subsection, we will compare the classification performance among ELM, BP, and SVM classifiers. All the experiments for the algorithms are carried out in MATLAB 8.0 environment running in an Inter i5, 2.6 GHz CPU. There are many variants of BP and SVM algorithms. Levenberg Marquardt BP (LM-BP) and least-square SVM (LS-SVM) [28] are used in this paper.

The data are randomly divided into two sets, namely, the training set and the testing set. The classification process

consists of two parts, generating the model by the training set and testing the performance of the model by the testing set. In our experiments, all the inputs (attributes) have been normalized into the range  $[0, 1]$ . Table 1 shows the confusion matrices between the 6 classes of activities (including sitting, standing, walking, running, going upstairs, and going downstairs) for the algorithms. The row represents the actual class and the column represents the recognized class by the algorithms, and the element at the  $i$ th row and  $j$ th column of the confusion matrix represents the probability of the actual class  $i$  is recognized as class  $j$  by an algorithm. The probability was calculated by 1000 samples over the window of six activities. From the matrices, we can find that ELM can distinguish the 6 classes of activities better than LM-BP and LS-SVM in general, and the KDA implementation on original features can improve the classification performance.

Furthermore, 50 trials have been conducted for a thorough comparison study and the average results are outputted as the classification results as shown in Table 2. The classifier of KDA based ELM takes the shortest testing time (0.0012 s) and achieves the highest testing accuracy (99.81%). Although KDA based ELM takes slight more training time than the original ELM, it has better classification result than the original ELM. That is because KDA can achieve the low intraclass variance and high cross class variance for activities and improve the classification accuracy effectively.

As shown in Table 2, KDA based LM-BP can classify samples faster than the original LM-BP and has higher classification accuracy. Also KDA based LS-SVM can improve the accuracy. We can find that the KDA strategy is useful in improving the recognition performance.

Now we give the comparison among ELM, BP, and SVM classifiers. It can be found from Table 2 that the classification accuracy of ELM, LM-BP, and LS-SVM based on the original features is not significantly different, but ELM has much faster learning speed (up to hundreds times) than LM-BP and LS-SVM. Based on the KDA implementation, ELM and LS-SVM can achieve better classification performance than LM-BP, but LS-SVM needs longer time for training than ELM. Also, the parameters of LS-SVM and LM-BP need to be determined before training, while ELM selects parameters randomly before training. Overall, the experiments demonstrate that ELM can achieve superior recognition performance compared with the LM-BP and LS-SVM classifiers.

#### 4. Conclusion

This paper develops a highly efficient approach for human activity recognition based on ELM and using only one triaxial accelerometer. A number of features, respectively, in the time domain, the frequency domain, and the time-frequency domain, are defined and extracted from the raw measurement signals. KDA is performed on the original features to achieve the low intraclass variance and high cross class variance for activities. ELM classifier is proposed to classify the activities. Experimental results show that KDA based classifier can improve the classification accuracy effectively and ELM can achieve superior recognition performance compared with SVM and BP classifiers.

TABLE 1: The confusion matrices between the 6 classes of activities. R indicates running, ST indicates sitting, SD indicates standing, W indicates walking, U indicates going upstairs, and D indicates going downstairs.

| (a) ELM based on original features    |       |       |       |       |       |       |
|---------------------------------------|-------|-------|-------|-------|-------|-------|
|                                       | R     | ST    | SD    | W     | U     | D     |
| R                                     | 1     | 0     | 0     | 0     | 0     | 0     |
| ST                                    | 0     | 1     | 0     | 0     | 0     | 0     |
| SD                                    | 0     | 0     | 0.999 | 0     | 0     | 0.001 |
| W                                     | 0     | 0     | 0     | 0.985 | 0.005 | 0.01  |
| U                                     | 0     | 0     | 0     | 0.015 | 0.968 | 0.017 |
| D                                     | 0     | 0     | 0     | 0.021 | 0.009 | 0.97  |
| (b) ELM based on KDA features         |       |       |       |       |       |       |
|                                       | R     | ST    | SD    | W     | U     | D     |
| R                                     | 1     | 0     | 0     | 0     | 0     | 0     |
| ST                                    | 0     | 1     | 0     | 0     | 0     | 0     |
| SD                                    | 0     | 0     | 1     | 0     | 0     | 0     |
| W                                     | 0     | 0     | 0     | 1     | 0     | 0     |
| U                                     | 0     | 0     | 0     | 0     | 0.999 | 0.001 |
| D                                     | 0     | 0     | 0     | 0.001 | 0     | 0.999 |
| (c) LM-BP based on original features  |       |       |       |       |       |       |
|                                       | R     | ST    | SD    | W     | U     | D     |
| R                                     | 0.998 | 0     | 0     | 0.002 | 0     | 0     |
| ST                                    | 0     | 0.998 | 0     | 0     | 0.002 | 0     |
| SD                                    | 0     | 0     | 0.996 | 0.003 | 0     | 0.001 |
| W                                     | 0     | 0     | 0     | 0.98  | 0.006 | 0.014 |
| U                                     | 0     | 0     | 0     | 0.018 | 0.962 | 0.02  |
| D                                     | 0     | 0     | 0.006 | 0.024 | 0.012 | 0.958 |
| (d) LM-BP based on KDA features       |       |       |       |       |       |       |
|                                       | R     | ST    | SD    | W     | U     | D     |
| R                                     | 1     | 0     | 0     | 0     | 0     | 0     |
| ST                                    | 0     | 1     | 0     | 0     | 0     | 0     |
| SD                                    | 0     | 0     | 1     | 0     | 0     | 0     |
| W                                     | 0     | 0     | 0     | 0.99  | 0.006 | 0.004 |
| U                                     | 0     | 0     | 0     | 0.01  | 0.988 | 0.002 |
| D                                     | 0     | 0     | 0     | 0.009 | 0.006 | 0.985 |
| (e) LS-SVM based on original features |       |       |       |       |       |       |
|                                       | R     | ST    | SD    | W     | U     | D     |
| R                                     | 0.993 | 0     | 0     | 0.006 | 0     | 0.001 |
| ST                                    | 0     | 0.996 | 0     | 0     | 0.004 | 0     |
| SD                                    | 0     | 0     | 0.997 | 0.001 | 0     | 0.002 |
| W                                     | 0     | 0     | 0     | 0.981 | 0.006 | 0.013 |
| U                                     | 0     | 0     | 0     | 0.01  | 0.976 | 0.014 |
| D                                     | 0     | 0     | 0     | 0.02  | 0.012 | 0.968 |
| (f) LS-SVM based on KDA features      |       |       |       |       |       |       |
|                                       | R     | ST    | SD    | W     | U     | D     |
| R                                     | 1     | 0     | 0     | 0     | 0     | 0     |
| ST                                    | 0     | 1     | 0     | 0     | 0     | 0     |
| SD                                    | 0     | 0     | 1     | 0     | 0     | 0     |
| W                                     | 0     | 0     | 0     | 0.998 | 0     | 0.002 |
| U                                     | 0     | 0     | 0     | 0.003 | 0.993 | 0.004 |
| D                                     | 0     | 0     | 0     | 0.002 | 0.002 | 0.996 |

TABLE 2: Classification results of ELM, LM-BP, and LS-SVM classifiers.

| Features  | Classifier | Time (s)      |               | Accuracy rate (%) |              |
|-----------|------------|---------------|---------------|-------------------|--------------|
|           |            | Training      | Testing       | Training          | Testing      |
| Original  | ELM        | <b>0.0022</b> | 0.0031        | 96.44             | 96.07        |
| KDA based | ELM        | 0.0031        | <b>0.0012</b> | <b>99.96</b>      | <b>99.81</b> |
| Original  | LM-BP      | 1.8985        | 0.0175        | 95.59             | 95.24        |
| KDA based | LM-BP      | 0.8861        | 0.0246        | 98.48             | 98.02        |
| Original  | LS-SVM     | 0.7775        | 0.1819        | 97.36             | 95.97        |
| KDA based | LS-SVM     | 0.7678        | 0.2760        | 99.94             | 99.05        |

Further research is required to determine the most appropriate feature set for more specific subject groups, such as the elderly or the neurologically impaired. In the future, we would also like to examine more complex physical activities and apply the ELM algorithm to solve more complex classification problems.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgments

The authors thank Professor Jiankang Wu for his valuable comments, Jian Cui, Peiyuan Chen, and Yuejie Lu for their work related to this research issue, and Dr. Lianying Ji for his insightful suggestions.

### References

- [1] R.-F. Li, L.-L. Wang, and K. Wang, "A survey of human body action recognition," *Pattern Recognition and Artificial Intelligence*, vol. 27, no. 1, pp. 35–48, 2014.
- [2] Q. R. Sun, W. M. Wang, and H. Liu, "Study of human action representation in video sequences," *CAAI Transaction on Intelligent Systems*, vol. 8, pp. 189–198, 2013.
- [3] N. Kern, B. Schiele, and A. Schmidt, "Multi-sensor activity context detection for wearable computing," in *Ambient Intelligence*, vol. 2875 of *Lecture Notes in Computer Science*, pp. 220–232, Springer, Berlin, Germany, 2003.
- [4] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*, vol. 3001 of *Lecture Notes in Computer Science*, pp. 1–17, Springer, Berlin, Germany, 2004.
- [5] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN '06)*, pp. 113–116, April 2006.
- [6] M. J. Mathie, A. C. F. Coster, N. H. Lovell, and B. G. Celler, "Detection of daily physical activities using a triaxial accelerometer," *Medical & Biological Engineering & Computing*, vol. 41, no. 3, pp. 296–301, 2003.
- [7] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim, "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1166–1172, 2010.
- [8] M. Li, V. Rozgić, G. Thatté et al., "Multimodal physical activity recognition by fusing temporal and cepstral information," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 4, pp. 369–380, 2010.
- [9] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 3, pp. 871–879, 2009.
- [10] T. Ishii and S. Abe, "Feature selection based on kernel discriminant analysis for multi-class problems," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '08)*, pp. 2455–2460, June 2008.
- [11] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, July 2004.
- [12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [13] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [14] G. B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.
- [15] E. Cambria, G. B. Huang, L. L. C. Kasun et al., "Extreme learning machines [trends & controversies]," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.
- [16] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the IEEE 9th Conference on Industrial Electronics and Applications (ICIEA '14)*, pp. 1163–1168, Hangzhou, China, June 2014.
- [17] W. Xiao, P. Liu, W.-S. Soh, and G.-B. Huang, "Large scale wireless indoor localization by clustering and extreme learning machine," in *Proceedings of the 15th International Conference on Information Fusion (FUSION '12)*, pp. 1609–1614, September 2012.
- [18] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.

- [19] W. Xiao, P. Liu, W.-S. Soh, and Y. Jin, "Extreme learning machine for wireless indoor localization," in *Proceedings of the 11th ACM/IEEE Conference on Information Processing in Sensing Networks (IPSN '12)*, pp. 101–102, April 2012.
- [20] H. G. Zhang, S. Zhang, and Y. X. Yin, "An improved ELM algorithm based on EM-ELM and ridge regression," in *Proceedings of the International Conference on Intelligence Science and Big Data Engineering*, Lecture Notes in Computer Science, pp. 756–763, Springer, 2013.
- [21] H. G. Zhang, S. Zhang, and Y. X. Yin, "A novel improved ELM algorithm for a real industrial application," *Mathematical Problems in Engineering*, vol. 2014, Article ID 824765, 7 pages, 2014.
- [22] J. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [23] D. Wang, D. Li, and Y. Lin, "A new method of face recognition with data field and PCA," in *Proceedings of the IEEE International Conference on Granular Computing (GrC '13)*, pp. 320–325, December 2013.
- [24] A. Iosifidis, A. Tefas, and I. Pitas, "Activity-based person identification using fuzzy representation and discriminant learning," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 530–542, 2012.
- [25] Z. A. Khan and W. Sohn, "Hierarchical human activity recognition system based on R-transform and nonlinear kernel discriminant features," *Electronics Letters*, vol. 48, no. 18, pp. 1119–1120, 2012.
- [26] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT press, 2002.
- [27] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision*, vol. 2, pp. 1029–1036, December 2004.
- [28] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

## Research Article

# Real-Time and Accurate Indoor Localization with Fusion Model of Wi-Fi Fingerprint and Motion Particle Filter

Xinlong Jiang,<sup>1,2,3</sup> Yiqiang Chen,<sup>1,2</sup> Junfa Liu,<sup>1,2</sup> Dingjun Liu,<sup>4</sup>  
Yang Gu,<sup>1,2,3</sup> and Zhenyu Chen<sup>1,2,3</sup>

<sup>1</sup>Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing 100190, China

<sup>2</sup>Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

<sup>3</sup>University of Chinese Academy of Sciences, Beijing 100190, China

<sup>4</sup>Xiangtan University, Hunan 411105, China

Correspondence should be addressed to Yiqiang Chen; yqchen@ict.ac.cn

Received 24 August 2014; Revised 4 November 2014; Accepted 31 December 2014

Academic Editor: Tao Chen

Copyright © 2015 Xinlong Jiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As the development of Indoor Location Based Service (Indoor LBS), a timely localization and smooth tracking with high accuracy are desperately needed. Unfortunately, any single method cannot meet the requirement of both high accuracy and real-time ability at the same time. In this paper, we propose a fusion location framework with Particle Filter using Wi-Fi signals and motion sensors. In this framework, we use Extreme Learning Machine (ELM) regression algorithm to predict position based on motion sensors and use Wi-Fi fingerprint location result to solve the error accumulation of motion sensors based location occasionally with Particle Filter. The experiments show that the trajectory is smoother as the real one than the traditional Wi-Fi fingerprint method.

## 1. Introduction

Nowadays, indoor location and tracking are very important in our daily life. Indeed, many applications require timely and accurate location information of the mobiles (context-aware application, emergency situation. . .) [1, 2]. But Global Positioning System (GPS), Assisted GPS (AGPS), and Mobile Base Station (MBS) cannot be used in indoor areas, as the signals cannot reach indoor area.

Fortunately, now many buildings are equipped with WLAN Access Points (APs), such as office buildings, hospitals, shopping malls, museums, and airports. It becomes easy to offer indoor location services without any other infrastructure investment. A common idea of Wi-Fi based indoor location is fingerprint method [3–8]. It is a classical classification problem where different supervised machine learning techniques have been used. A lot of researches show that Wi-Fi signal can generate good prediction for indoor localization [9], but it will spend some time to scan the Wi-Fi signal around so that it cannot offer continuous location service.

Nowadays, the smartphone is the most wildly used device, in part due to its abundant inertial sensors, such as accelerometer, magnetometer and gyroscope. They can be used to measure people's motion state, including walking speed and orientation. Based on this information, we can estimate the relative changing of position by determining a pedestrian's movement. This kind of methods is based on the Inertial Navigation System (INS) [10] method, which can offer continuous localization. But due to its iterative process, we can only estimate the position with high accuracy in a short time. The error accumulation will lead the gap between the movement and the estimated result increases continuously.

By the way, the power consumption is also important. As far as we know, the battery usage of Wi-Fi and communication module is several dozen times higher than the Inertial Measurement Unit (IMU) that consists of gyroscopes and accelerometers. Carroll and Heiser [11] also did some analysis of power consumption in smartphones and the same conclusions can be seen.

Therefore, in this paper, we combine the advantages of two methods and present real-time and accurate indoor localization based on fusion model of Wi-Fi fingerprint and motion Particle Filter. We take use of the accurate Wi-Fi location result to modify the motion Particle Filter in the correction step, so to eliminate the error accumulation caused by motion sensors based location. And this system includes three main modules: sensors data based location model (ELM regression); Wi-Fi based location model (ELM +  $K$ -NN classification); and the fusion Particle Filter model.

The rest of the paper is organized as follows. We firstly review some related works of indoor location and tracking in Section 2. Then we give a brief introduction of proposed method in Section 3. Section 4 demonstrates the feature selection for sensors data. In Section 5, we introduce ELM algorithm. After that, we introduce Particle Filter for sensors based location in Section 6. In Section 7, we introduce the fusion model of Wi-Fi fingerprint and motion Particle Filter. Experiments and performance evaluation are given in Section 8. In the end, we make a short conclusion in Section 9.

## 2. Related Works

As the basic support of Indoor LBS application, a timely localization with high-accuracy and low power consumption is very important. At present, one of the most popular techniques is Wi-Fi fingerprint based location [3–8]. References [3–8] take use of the existing wireless network infrastructures to avoid extra deployment costs.

Wi-Fi fingerprint based locations contain two phrases: offline training and online mapping. During the offline phase, a fingerprint information map will be built. In this map, there are some registered points and corresponding Received Signal Strength (RSS) from the APs around. On the online phase, people receive the Wi-Fi signals, and then some machine learning algorithms will be used to predict the position based on fingerprint dataset. It becomes a classical classification problem where different supervised machine learning techniques have been used to train classifiers, using the signal strength from different APs as the feature and providing the location estimation as their output estimation. Nearest Neighbor ( $K$ -NN), Decision Tree (DT), Bayesian, Support Vector Machine (SVM), and ELM [12] are most frequently used by location fingerprint. Among all the algorithms above, ELM is more and more widely used for its competitive fast learning speed during both offline and online phrases.

Although Wi-Fi signals can generate good prediction model for indoor position estimation, it will take a while to scan the Wi-Fi signals so that it cannot offer continuous location service. As shown in Figure 1, we select four different android smartphones (HTC G12, Sony LT26ii, HTC G10, and Xiaomi 1S). With each smartphone, we scan the Wi-Fi for 10 times (Test. 1–10). We can conclude that every time we collect the Wi-Fi signals, it will take more than 1 second on average. That is to say, Wi-Fi based location cannot offer immediately location result due to the time consumption of signals scanning.

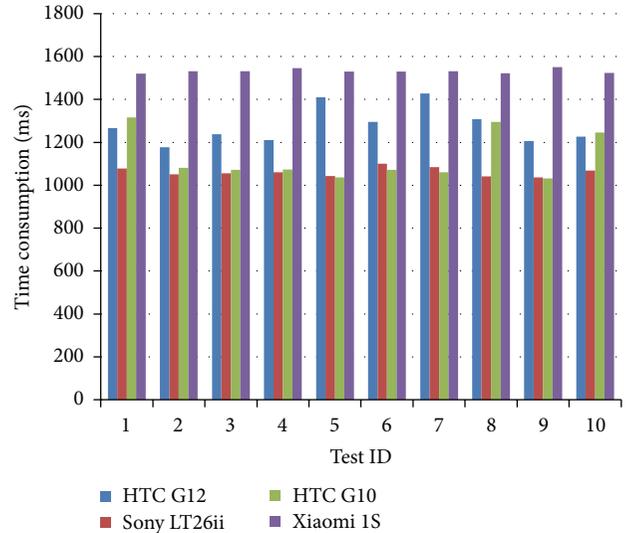


FIGURE 1: Time consumption of Wi-Fi signal scanning.

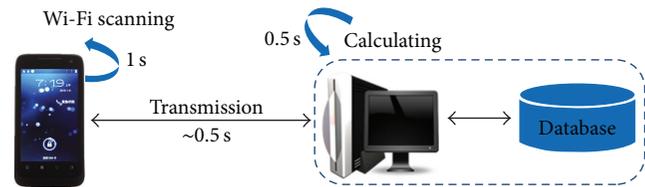


FIGURE 2: Location system with C-S (Client-Server) architecture.

According to the experiments on Wi-Fi based location system developed by our lab, shown in Figure 2, we can get one location result in every 2 seconds including some other time consumption of transmission and calculation.

In order to overcome the shortcoming of Wi-Fi based location, we can take other measures which can offer continuous location service. Nowadays, as a lot of sensors are integrated in smart phones, so that we can use them to measure people's motion state, including distance and orientation. Distance estimation mainly based on pedestrian model, including step detection and a step model. Step detection typically relies on peak detection over accelerometer [13]. But the result is not good enough, because it is sensitive to noise and other irrelevant motions, producing a high rate of false positives. And there are also some other methods that rely on detecting features such as zero-cross point [14]. Orientation is relatively easier to get via magnetometer, gyroscope, and accelerometer data so that we can get user heading orientation information with high accuracy easily. The orientation of smartphones themselves can be obtained from the motion sensors Application Programming Interface (API) [15] or computed from the raw sensors data.

But based on motion sensors, we can only estimate the position with high accuracy in a short time; it is difficult to locate independently for a long time period because of the error accumulation. To solve this problem, a lot of previous works have been done. Li et al. [16] not only developed an algorithm for reliable detection of steps, heading directions,

and accurate estimation of step length, but also built an end-to-end location system integrating these models. But they still did not solve the problem of location error accumulation. Evennou et al. [1] applied a developed Particle Filter method to WLAN location determination technique and proposed an indoor mobile positioning system. They used the Particle Filter and its constraint on a Voronoi diagram to represent an interesting, so as to handle the variations of the signal strength measurements. It can also aggregate different information like signal strength and the map to obtain correct trajectories without wall-crossings. But in real applications, it is difficult to get the geography Information.

From the related works above, we can see that Wi-Fi based location can offer high accuracy, but it needs high time and power consumptions. Inertial sensors based location estimates the position with high accuracy in a short time, but the shortage is the error accumulation. Thus, in this paper, we present real-time and accurate indoor location based on fusion model of Wi-Fi fingerprint method and motion Particle Filter.

### 3. Overview of Our Method

The architecture of our method is shown in Figure 3. It contains three main parts. (1) Sensors data based short-time location with ELM regression: we can determine the orientation and velocity to tracking users, but as every location result is based on the former one, it will cause error accumulation. (2) Wi-Fi based location with ELM classification and K-NN: we can get high location accuracy with this method, but as the time consumption of Wi-Fi signals scanning and data transmission, we can only get discrete result in every several seconds. (3) Fusion Particle Filter model: in order to solve the problems in the two methods mentioned above, we add a fusion Particle Filter model to eliminate the error accumulation caused by motion sensors based location with Wi-Fi location result in the correction step and offer a smooth location trajectory.

### 4. Feature Selection for Sensors Data

Currently, there are a lot of sensors integrated in smartphones. Some previous methods only directly use the API data, such as orientation. But sometimes, the API data cannot offer high precision. As shown in Figure 4, the orientation directly offered by API has a big bias with the real one caused by the environment changing. So in this paper, we use regression learning algorithm with a variety of sensors data to determine the motion state.

In order to find the strong correlation between movement state and sensors data, we investigate the existing sensors on the smartphone [15]. Generally, there are seven sensors that can reflect motion state, shown in Table 1.

When we mention localization, we want to get the orientation and velocity from sensors data. Among all the sensors above, the accelerometer, magnetic field, orientation, gyroscope, and gravity sensors have correlation with the orientation and velocity. We tried different combinations of

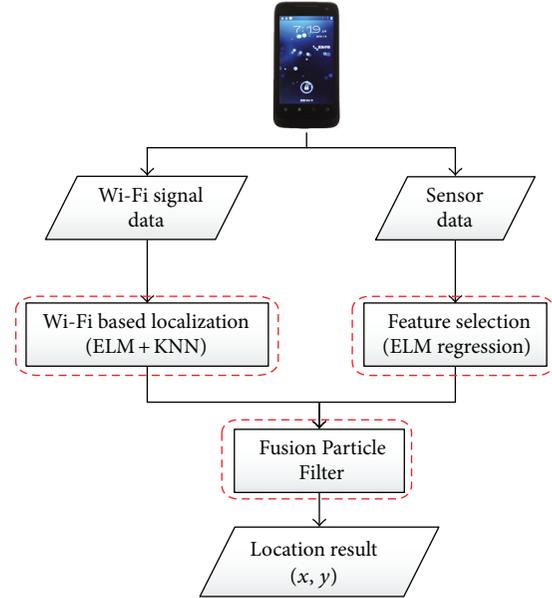


FIGURE 3: Architecture of our proposed method.

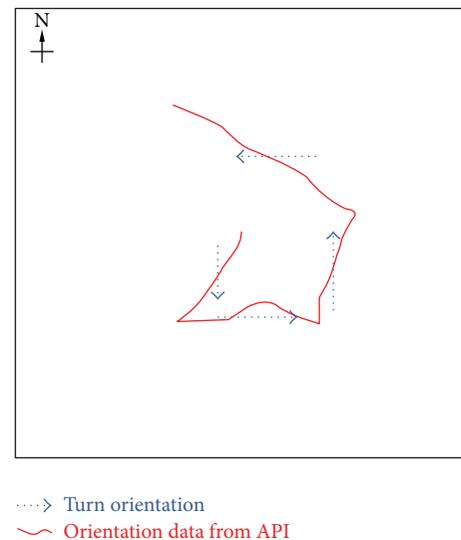


FIGURE 4: Orientation API output.

them to predict the orientation and velocity. Finally, we find that velocity has a strong relationship with accelerometer  $x$ ,  $y$ , and  $z$  axes and gyroscope  $x$ ,  $y$ , and  $z$  axes, as seen in the following formula:

$$v \sim \{acc.x, acc.y, acc.z, gyr.x, gyr.y, gyr.z\}. \quad (1)$$

And we use orientation  $x$ ,  $y$ , and  $z$  axes, magnetic field  $x$ ,  $y$ , and  $z$  axes as the features to determine orientation, as seen in the following formula:

$$o \sim \{ori.x, ori.y, ori.z, mag.x, mag.y, mag.z\}. \quad (2)$$

The reason why we choose  $\{acc.x, acc.y, acc.z, gyr.x, gyr.y, gyr.z\}$  as the feature to predict  $v$  is that different walking speed

TABLE 1: Motion sensors in smart phone.

| ID | Sensor              | # descriptions                                                                                          | # units of measure |
|----|---------------------|---------------------------------------------------------------------------------------------------------|--------------------|
| 1  | Accelerometer       | Acceleration force along the $x$ , $y$ , and $z$ axes                                                   | $\text{m/s}^2$     |
| 2  | Magnetic field      | Measures the ambient geomagnetic field for all the three physical axes ( $x$ , $y$ , and $z$ )          | $\mu\text{T}$      |
| 3  | Orientation         | Measures degree of rotation that a device makes around all three physical axes ( $x$ , $y$ , and $z$ ). | $^\circ$           |
| 4  | Gyroscope           | Rate of rotation around the $x$ , $y$ , and $z$ axes.                                                   | $\text{rad/s}$     |
| 5  | Gravity             | Force of gravity along the $x$ , $y$ , and $z$ axes.                                                    | $\text{m/s}^2$     |
| 6  | Linear acceleration | Acceleration force along the $x$ , $y$ , and $z$ axes.                                                  | $\text{m/s}^2$     |
| 7  | Rotation vector     | Rotation vector component along the $x$ , $y$ , and $z$ axes.                                           | Unitless           |

causes different shaking degree, which can be observed from the accelerometer readings. As shown in Figure 5, we can conclude that this user walks faster and faster.

We use the same way to choose  $\{\text{ori}.x, \text{ori}.y, \text{ori}.z, \text{mag}.x, \text{mag}.y, \text{mag}.z\}$  as the feature vector to predict orientation  $o$ . Although orientation sensor's output cannot match the real orientation very well, it can still offer some positive effect. Magnetic field measures the ambient geomagnetic field which has strong relationship with orientation.

In order to use machine learning algorithm to predict velocity and orientation, we firstly prepare training data.

Firstly, experiment participants walk along a fixed route with a smart phone. Motion sensor data will be collected, and participants will label the time at given points. The frequency of sensors is different in each smartphone. In our experiments  $f_s \approx 65 \text{ Hz}$ .

Secondly, we use Cubic Spline Interpolation [17] to smooth the trajectory.

Thirdly, we use slide-window to segment the trajectory with window length  $T_w$ . The  $T_w$  will be determined in the experiment, but we must keep  $T_w > 1/f_s$  so that in every single slide-window there is at least one integrated data. Let us take  $x$  component of acceleration in  $i$ th window as an example; we use the average of the sensors data as the feature according to

$$\text{acc}_{i,x} = \frac{1}{N_i} \sum_{k=1}^{N_i} \text{acc}_{ik,x}, \quad (3)$$

where  $N_i$  is the data number collected by the smartphone.

According to the smooth trajectory, in the  $i$ th stage, we can measure the orientation  $o_i$  and velocity  $v_i$ , which can be treated as the label for regression algorithm.

## 5. ELM Algorithm

After feature selection, we are going to predict the motion state. Commonly, the accelerometer has been used to count the number of steps. Then estimate the distance with the user's steps stride [18]. But it depends on the assumption that user's steps strides are always the same, which is hard to satisfy in practical applications.

To avoid this assumption, we adopt ELM to train an effective model which can offer real-time prediction of orientation and walking speed.

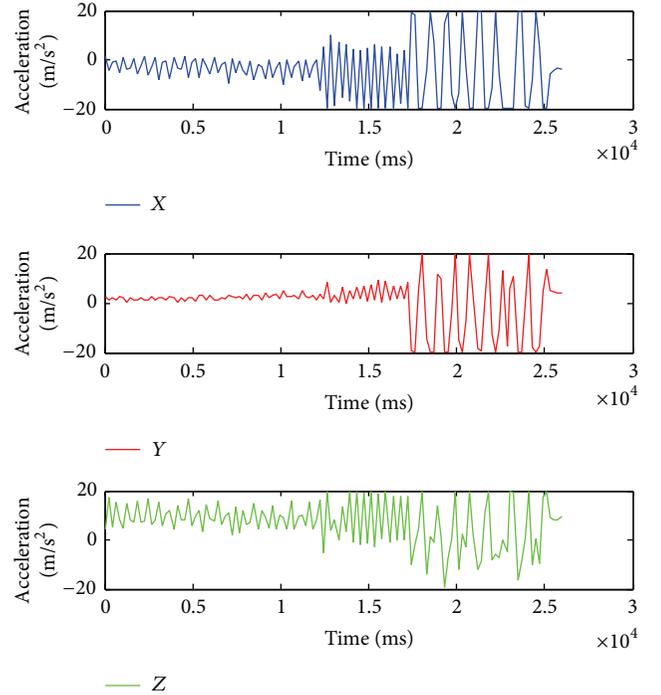
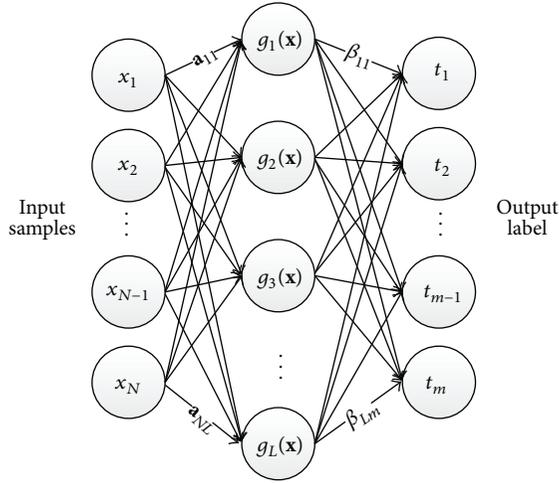


FIGURE 5: Accelerometer and walking speed.

ELM [12] is an Artificial Neural Network (ANN), especially Single Layer Feedforward Networks (SLFN), developed by Huang et al. Given  $N$  arbitrary distinct samples  $(\mathbf{x}_i, \mathbf{t}_i) \in R^n \times R^m$ ,  $i = 1, 2, \dots, N$ . Here,  $\mathbf{x}_i$  is an  $n \times 1$  input vector  $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$  and  $\mathbf{t}_i$  is an  $m \times 1$  target vector  $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T$ . The network with  $L$  hidden nodes is shown in Figure 6. The output function of this network can be represented as follows:

$$f_L(\mathbf{x}_j) = \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j), \quad j = 1, \dots, N, \quad (4)$$

where  $\mathbf{a}_i$  and  $b_i$  are the learning parameters of hidden nodes and  $\beta_i$  is the weight connecting the  $i$ th hidden node to the output node.  $G(\mathbf{a}_i, b_i, \mathbf{x})$  is the output of the  $i$ th hidden node with respect to the input  $\mathbf{x}$ . For additive hidden node with


 FIGURE 6: SLFN with  $L$  hidden neurons.

the activation function  $g(x) : R \rightarrow R$  (e.g., sigmoid and threshold),  $G(\mathbf{a}_i, b_i, \mathbf{x})$  is given below:

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad b_i \in R. \quad (5)$$

If an SLFN with  $L$  hidden nodes can approximate these  $N$  samples with zero errors, this then implies that there exist  $\beta_j$ ,  $\mathbf{a}_i$ , and  $b_i$  such that

$$f_L(\mathbf{x}_j) = \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \dots, N. \quad (6)$$

Equation (6) can be summarized as

$$H\beta = T, \quad (7)$$

where

$$H(\mathbf{a}_1, \dots, \mathbf{a}_L, b_1, \dots, b_L, \mathbf{x}_1, \dots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \dots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \dots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}, \quad (8)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \quad T = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}. \quad (9)$$

According to [12], the hidden node parameters  $\mathbf{a}_i$  and  $b_i$  (input weights and biases or centers and impact factors) of SLFNs need not be tuned during training and may simply be assigned with random values. Equation (7) then becomes a linear system and the output weights  $\beta$  are estimated as

$$\hat{\beta} = H^\dagger T = (H^T H)^{-1} H^T T, \quad (10)$$

where  $H^\dagger$  is the Moore-Penrose generalizes inverse of hidden layer output matrix  $H$ . Equation (10) can be seen as a linear system; the least-squares solution of (8) is the answer of ELM.

By now, ELM and some other related algorithms such as OS-ELM [19, 20], SELM [21, 22], and Unsurprised-ELM [22] have been developed and widely applied in some real applications. Huang [23] made a further insight into ELM, including some discussion about random neurons, random features, and kernels. And in [24, 25], Cao et al. made some improvement to the basic ELM and achieved a good performance. Because of the fast learning speed and good learning ability, ELM has been widely used in ubiquitous computing applications, such as activity recognition [26–29]. That is why we use ELM model both in offline learning and in online prediction. Our lab has done a lot of Wi-Fi indoor location research based on it and achieved good performance [21, 30].

From all the information above, we can see that ELM model can be used in regression and classification problems. In our application, the velocity and orientation are all continuous variable. So we use ELM as a regression model.

(i) *ELM Model for Velocity*. For velocity, the input vector is a vector with six features:

$$\mathbf{x}_i = [\text{acc}_i.x, \text{acc}_i.y, \text{acc}_i.z, \text{gry}_i.x, \text{gry}_i.y, \text{gry}_i.z]^T \quad (11)$$

and the output vector is only a scalar value  $\mathbf{t}_i = [v_i]^T$ .

(ii) *ELM Model for Orientation*. For orientation, the input is a vector with four features:

$$\mathbf{x}_i = [\text{ori}_i.x, \text{ori}_i.y, \text{ori}_i.z, \text{mag}_i.x, \text{mag}_i.y, \text{mag}_i.z]^T \quad (12)$$

and the output vector is also a scalar value  $\mathbf{t}_i = [o_i]^T$ .

According to [31], we also have offline and online phases. During offline phases, we will collect the sensors data and the trajectory to train these two models mentioned above. During the online phases, we will use sensors data and models to determine  $v$  and  $o$ . Then we can use (13) to calculate the current position. Consider

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{cases} \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} T_w & 0 \\ 0 & T_w \end{bmatrix} * \begin{bmatrix} v_t & 0 \\ 0 & v_t \end{bmatrix} * \begin{bmatrix} \sin(o_t) \\ \cos(o_t) \end{bmatrix}, & t > 1, \\ \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, & t = 0. \end{cases} \quad (13)$$

As our method is based on an iterative process, so the initial position  $[x_0 \ y_0]^T$  should be given.

By now, we can already track the trace, given the initial position, in every  $T_w$  second; we collect the sensors data and

use these two ELM models to determine  $v$  and  $o$ . So that, the current position will be calculate by (13). But unfortunately, sensors on smart phone are not accurate enough to offer data with high confidence. For a short time, it is really a reliable method, but as time goes on, error accumulation will destroy the reliability.

## 6. Particle Filter for Sensors Based Location

Particle Filter [32] is an on-line posterior density estimation algorithm estimating the posterior density of the state-space by directly implementing the Bayesian recursion equations. It provides a well-established methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions. The state-space model can be nonlinear and the initial state and noise distributions can take any form required. For a lot of position and navigation problems, common motion model based filters can be applied. Models that are linear in the state dynamics and nonlinear in the measurements are considered [33].

*State Transition Equation.* Consider

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}_u\mathbf{u}_t + w_t. \quad (14)$$

*Observation Equation.* Consider

$$\mathbf{y}_t = h(\mathbf{x}_t) + e_t, \quad (15)$$

where  $\mathbf{x}_t$ : state vector,  $\mathbf{u}_t$ : measured inputs,  $w_t$ : process noise,  $\mathbf{y}_t$ : measurements, and  $e_t$ : measurement error.

In our application, we turn the model, (14) and (15), into a specific one, as shown in

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} T_w & 0 \\ 0 & T_w \end{bmatrix} \begin{bmatrix} v_{xt} \\ v_{yt} \end{bmatrix} + Q, \quad (16)$$

$$\begin{bmatrix} X_t \\ Y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + R. \quad (17)$$

In (16),  $[x_t, y_t]^T$  denotes the state vector associated to each particle (position).  $T_w$  is the elapsed time between the  $(t-1)$ th and the  $t$ th measurements.  $Q$  is the process noise. Here the used process is a zero mean Gaussian noise with a  $0.1 \text{ m/s}^2$  variance which is a realistic model of pedestrian movement.

Equation (17) is observation equation, which denotes the relationship between the hidden states  $[x_t, y_t]^T$  and the observation states  $[X_t, Y_t]$ . Because we directly observe the coordinate point from ELM regression algorithm, there is no measurement error in this process, so we set  $R$  to be 0.

A Particle Filter approximates the position at time  $t$  by a set of  $N$  particles. Each particle contains a position  $z_k^i$  and a weight  $P_r[x_t | z_k^i]$ .  $P_r[x_t | z_k^i]$  indicates the importance of the  $i$ th particle. In the case of an indoor movement, the following law has been retained:

$$P_r[x_t | z_k^i] = \frac{1}{\sqrt{2\pi\sigma}} \exp \left[ -\frac{(X_{xt} - X_{z_k^i})^2}{2 \cdot \sigma^2} \right]. \quad (18)$$

With  $X_{xt}$  the position returned by the ELM regression algorithm,  $X_{z_k^i}$  the position of the  $i$ th particle at time  $t$ , and  $\sigma$  the measurement confidence. The smaller  $\sigma$  will be, the more confident the user is in the measurement. That would mean that there is very little variation in the measurements for the same position. Here,  $\sigma = 50$  was chosen.

According to the process of Particle Filter algorithm, there are four steps in a circle: prediction, correction, particle update, and resampling. At each  $T_w$  second, we will determine a new position by the following steps.

### (i) Feature Selection

- (1) We collect sensors data from mobile and calculate the feature vector according to (3).
- (2) Then we use two ELM regression models to calculate the orientation  $o_i$  and velocity  $v_i$ .

### (ii) Then We Go to Particle Filter Phase

- (3) Prediction: during this step, every particle will propagate to a new one based on state transition equation (16). After that, all the particles move to new places.
- (4) Correction: now, we have all the particles and we must give them the weight; according to  $o_i$ ,  $v_i$ , and location of last moment, we can calculate the observed value by (17). Then we can use (18), from which we can see that the more particle far away from the observed value, the less value the particle has.
- (5) Particle Update: the weight update equation is given in [33, 34]:

$$w_t^i = w_{t-1}^i \cdot P_r[x_t | z_t^i]. \quad (19)$$

To obtain the posterior density function, we have to normalize those weights. After a few iterations, only a few particles can still alive. So we should go to resampling step to avoid having few remaining particles.

- (6) Resampling: the resampling step is the critical point for the PF. The fundamental idea is to remove the particles which have too low weight. Various resampling algorithms can be chosen. We choose the Sequential Importance Resampling (SIR) for the reason that it is a simple one which is wildly used in SLAM (Simultaneous Location and Mapping) [35]. Take  $N$  samples with replacement from the set  $\{x_t^i\}_{i=1}^N$ , where the probability to take  $i$ th sample is  $w_t^i$ . Let  $w_t^i = 1/N$ .
- (7) Finally, we can calculate the new location result by  $\hat{x}_t = \sum_{i=1}^N w_t^i x_t^i$ . And then go to step (1) until stop.

## 7. Fusion Model of Wi-Fi Fingerprint and Motion Particle Filter

By now, only the sensors data are used for localization. But they cannot offer long time location service with high accuracy due to the error accumulation. To solve this problem,

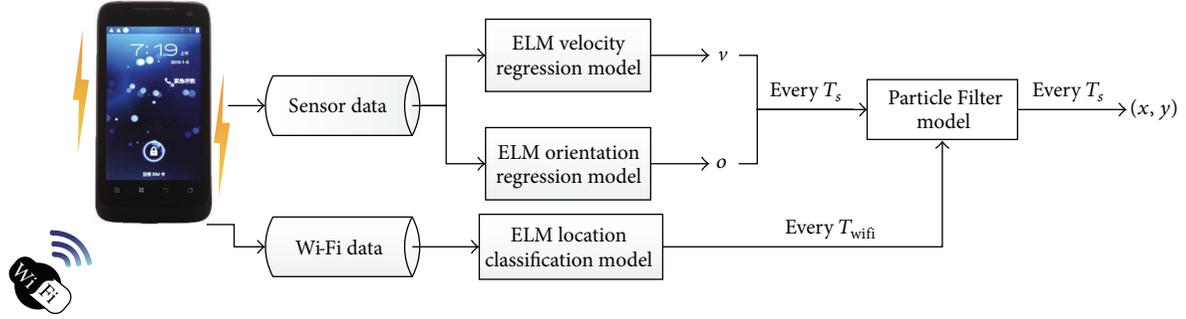


FIGURE 7: Location process of the whole system.

we present a fusion Particle Filter model, combining the advantages of motion sensors based location and Wi-Fi based location.

Wi-Fi fingerprint based location methods include offline learning and online prediction. During the offline phase, we collect fingerprints, including coordinate  $\{x, y\}$  and Wi-Fi Received Signal Strength Indication (RSSI)  $\{r_1, r_2, \dots, r_k\}$  measured at  $\{x, y\}$ . Actually, we do not know the total number of Wi-Fi Aps and their position. We firstly calibrate all the location area and then choose the several most frequent APs as the feature. After that, we change all the fingerprints to only contain these frequent APs and supplement the missing items with default value  $-95$  dB.

According to [36], during the online phase, we use ELM classification and  $K$ -NN algorithm.

With ELM, we can find the  $K$  Nearest Neighbor points, but we do not know the distance because the ELM only offers the nearest points but not the weights. So we use the Euclidean distance of signal strength to determine the weight according to

$$\begin{aligned} \text{dis}_k &= \sqrt{\sum_{i=1}^n (\text{rss}_{ki} - \text{rss}_i')^2}, \\ w_k &= \frac{\sum_{i=1}^K \text{dis}_i - \text{dis}_k}{(n-1) \sum_{i=1}^K \text{dis}_i}. \end{aligned} \quad (20)$$

When we use smart phone to do Wi-Fi based indoor location, it takes a little while to collect Wi-Fi data. So we can only get a location result in every  $T_{\text{Wi-Fi}}$  second. To eliminate the error accumulation caused by sensors based location, we add the Wi-Fi location result in particle update step and resampling step of Particle Filter.

As shown in Figure 7, once the smartphone begins to locate, the Wi-Fi signals and motion sensors data will be collected. By using the motion sensors data, we get a location result in every 200 milliseconds. And Wi-Fi based location data will arrive in about every 2 seconds.

We promote the Particle Filter mentioned in last section to a fusion one by adding the Wi-Fi location result in the correction step. When we go to the correction step, we should check whether there comes a Wi-Fi location result. If yes, we use the Wi-Fi location point as the observation value to calculate all particles' weight. And if not, we still use the result

TABLE 2: Mobile phones' hardware parameters.

|                   | # OS            | # CPU (Hz) | # RAM  | # ROM  |
|-------------------|-----------------|------------|--------|--------|
| HTC Desire S      | Android, v2.3   | 1 G        | 768 MB | 1 GB   |
| Samsung Galaxy S3 | Android, v4.0   | 1.4 G      | 1 GB   | 16 GB  |
| HTC Desire S      | Android, v2.2   | 1 G        | 768 MB | 1.5 GB |
| SONY Lt26ii       | Android, v4.0   | 1741 M     | 1 GB   | 32 GM  |
| HUAWEI S8600      | Android, v2.3.4 | 624 M      | 512 MB | 512 MB |

predicted by motion sensors data. With the fusion model, our system can offer a highly accurate location result with limited time.

## 8. Experiments and Performance Evaluation

In this section, we do several experiments to evaluate the performance of our method. All the experiments are running on smartphones and a computer with following configuration.

*Smart Mobile Phones.* See Table 2.

*Computer*

Operation System: Windows XP Professional SP3.

CPU: Intel Pentium(R) 4 CPU.

Main frequency: 3.2 GHz.

RAM: 2 G.

*8.1. Experimental Design and Data Preparing.* In order to evaluate the performance, we build a test platform in our work space, as shown in Figure 8. Including the aisle, it is about  $100 \text{ m}^2$  ( $8.5 \text{ m} * 12 \text{ m}$ ).

For the reason that we apply ELM algorithm in both Wi-Fi based location and sensor based location, we will collect data to train the model during offline phase.

Concerning sensors data, we plan a fixed route including straights and curves, shown in Figure 9. People walk along this trajectory with any speed: slow, middle, fast, and even run. And they record the time at every label point by touching the volume key. As shown in Figure 10. In order that we can measure different types of motion, there are four persons doing the collection work, and each person walks five times. Every smart phone will be used once by each person.



FIGURE 8: Our working space in ICT F8.

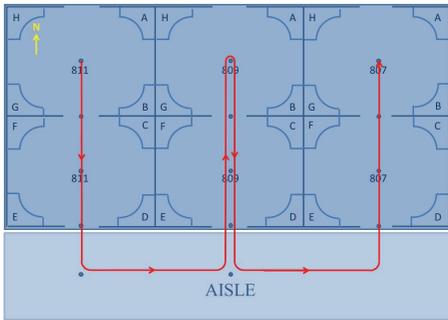


FIGURE 9: The presetting route.



FIGURE 10: Collecting the sensors data.

After collecting the original sensors data, we use Cubic Spline Interpolation to smooth the trajectory. Thirdly, we use slide-window to segment the trajectory with window length  $T_w$ , shown in Figure 11. The  $T_w$  will be determined in the experiment, but we must make sure  $T_w > 1/f_s$  so that in every single slide-window there is independent data. In every window, we will calculate feature by (1). And measure the orientation and speed from the red points.

For the Wi-Fi based location part, we use the fingerprint method. So we also have to build a signal distribution map. We select 36 points that can cover the test area, shown in Figure 12. Between every two adjacent points, it is about 1.5 m~2 m. We do not strictly require label points equally distributed in this area. We register four times at each point, collect Wi-Fi data for three times, and then use the mean value.

TABLE 3: MSE of orientation and velocity in different window length.

|                             | Window length |         |         |         |         |
|-----------------------------|---------------|---------|---------|---------|---------|
|                             | 200 ms        | 400 ms  | 600 ms  | 800 ms  | 1000 ms |
| MSE of orientation (degree) | 25.0529       | 27.8570 | 26.8727 | 26.5759 | 26.4503 |
| MSE of velocity ( $m/s^2$ ) | 0.3001        | 0.4020  | 0.4158  | 0.4642  | 0.4435  |

During the online phase, people were walking along the trajectory, shown in Figure 9, to collect testing data. We let a user to carry a smart phone to collect sensors data and get the Wi-Fi location result from the server. Then we will use these raws to predict the trajectory. After that, we can evaluate the performance by comparing with groundtruth.

## 8.2. Parameters Selection

**8.2.1. Determine the Time Window Length  $T_w$ .** Firstly, we will determine the window length  $T_w$ . For the reason that the sensors frequency is  $f_s \approx 65$  Hz. In order to void the influence caused by noise, we use the mean value as the features. So we should ensure that there are about ten samples in each window. That is to say the  $T_w$  should be bigger than 160 milliseconds.

We use Mean Square Error (MSE) [37] to measure the accuracy in ELM regression algorithms. The results are shown in Table 3. We can see that we can get lowest MSE in both orientation and velocity. So we set  $T_w$  to be 200 ms.

**8.2.2. Parameters of ELM Regression Models.** For basic ELM, we only need to determine the number of hidden nodes  $L$ . The input weights and biases of hidden nodes can be generated randomly. We have collected 5316 training samples and divided them into 10 parts. Then we use 10-fold cross validation to select best parameter; the results are shown in Figure 13.

From Figure 13, we can come to conclusions that, for the ELM regression model of velocity, 500 hidden nodes are suitable and, for orientation regression, we can minimize test error when the model has 1000 hidden nodes. Finally, we set  $L$  to be 500 for velocity and 1000 for orientation. And we chose *Sigmoid* [38] function for orientation and *Hardlim* [39] function for velocity.

**8.3. Experimental Performance.** In this section, in order to validate the proposed method, the comparisons to the existing methods are necessary. As the proposed method contains Wi-Fi and sensors information and takes use of ELM and Particle Filter algorithms, comparison experiments with some related methods should be involved. We firstly consider the Wi-Fi based location only and tried four different methods ( $K$ -NN, ELM, ELM + Kalman filter, and ELM + Particle Filter). Then we measure the location error of some sensors based methods and proposed method. Also the error accumulation tests have been considered as it is the key factor of location accuracy. Finally, we give a statistical result of

TABLE 4: Comparison of the performance of the different methods.

|                | Sensors data<br>(gait + Particle Filter)<br>[40] | Sensors data<br>(ELM) | Sensors data<br>(ELM + Kalman filter) | Sensors data<br>(ELM + Particle Filter) | Sensors data + Wi-Fi<br>(ELM + Particle Filter) |
|----------------|--------------------------------------------------|-----------------------|---------------------------------------|-----------------------------------------|-------------------------------------------------|
| Mean error (m) | 1.542                                            | 0.609                 | 0.829                                 | 0.633                                   | 0.150                                           |

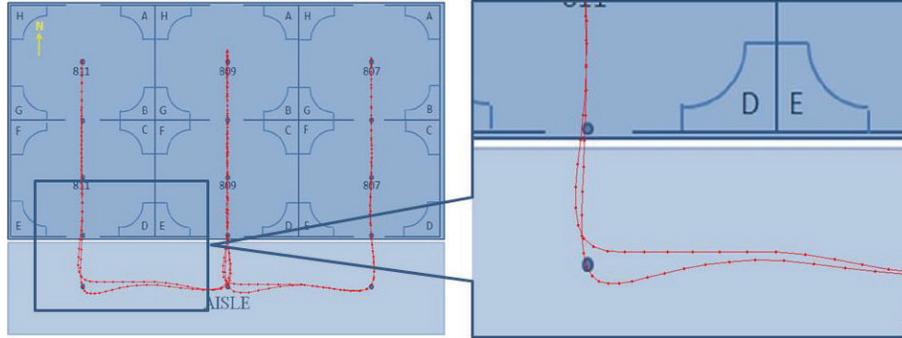


FIGURE 11: The trajectory after smooth and segmented by slide-window.

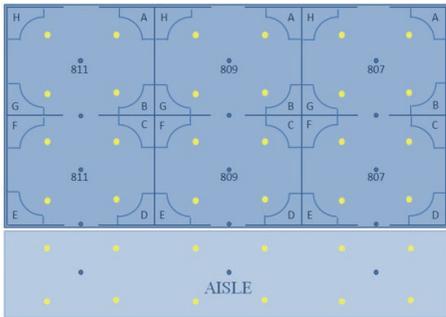


FIGURE 12: Fingerprint points are in yellow color.

the time consumption in our experiment to state that our proposed method is real practical.

**8.3.1. Evaluate the Location Accuracy.** Firstly, we compare the location performance of  $K$ -NN, ELM, Kalman filter, and Particle Filter based on Wi-Fi fingerprint. As shown in Figure 14, ELM performs better than  $K$ -NN, but it is still not very good as expected. We can explain the reason as that every location process is separated with each other. They are not treated as a sequence; thus some relevant information has been missed. So we apply Kalman filter and Particle Filter to the location results of ELM. Fortunately, both of them can raise the accuracy by big percentages. But, Particle Filter performs much better as it is a nonlinear system, which is more effective to simulate the walking pattern.

Besides the Wi-Fi information, sensors data is also very useful for indoor location. It is usually used to detect gait and then use a default step length to estimate the walking distance. In proposed method, we use ELM model to build a mapping

relationship of sensors data and walking pattern. Moreover, Kalman filter and Particle Filter also can be used to smooth the initial location results. To evaluate their performance, we not only compare the location mean error, but also calculate the location trajectory using Matlab. From Table 4, we can conclude that, comparing to gait detection based method mentioned in [40], ELM based methods can get better results. And with the Particle Filter, mean error becomes smaller. But we can see that when we apply the Kalman filter to the ELM location result, the mean error is bigger than before. We explain it as the function of Kalman filter is to make the trajectory smoother, not to minimize the mean error, which can be seen in Figure 14. But filters can only make the trajectory smoother; they cannot solve the error accumulation problem. In order to solve the error accumulation, we combine the Wi-Fi location with the sensors data based location to come into being proposed method, which indeed can offer a small mean error.

From Figure 15(a), we can see that trajectory of sensors based location is not very good. As time goes on, the trajectory is getting more and more far away from groundtruth. The reason is that, within a short period of time, the sensors can offer good valuation of walking orientation and speed, but the bias also exists. The next result is based on the previous step, so the bias will cumulate to unbearable error after a period of time. Figure 15(b) shows the trajectory of sensors based location with ELM and Kalman filter. We can see it making the trajectory smoother in some turnings. But it is a little more far from the groundtruth. We explain it as that the function of Kalman filter is to smooth the trajectory, but not to minimize the location error. Figure 15(c) indicates the sensors based location with ELM and Particle Filter. From it, we can conclude that the Particle Filter can make the trajectory smoother than before, but it does not improve the error accumulation by a large scale. The proposed method can

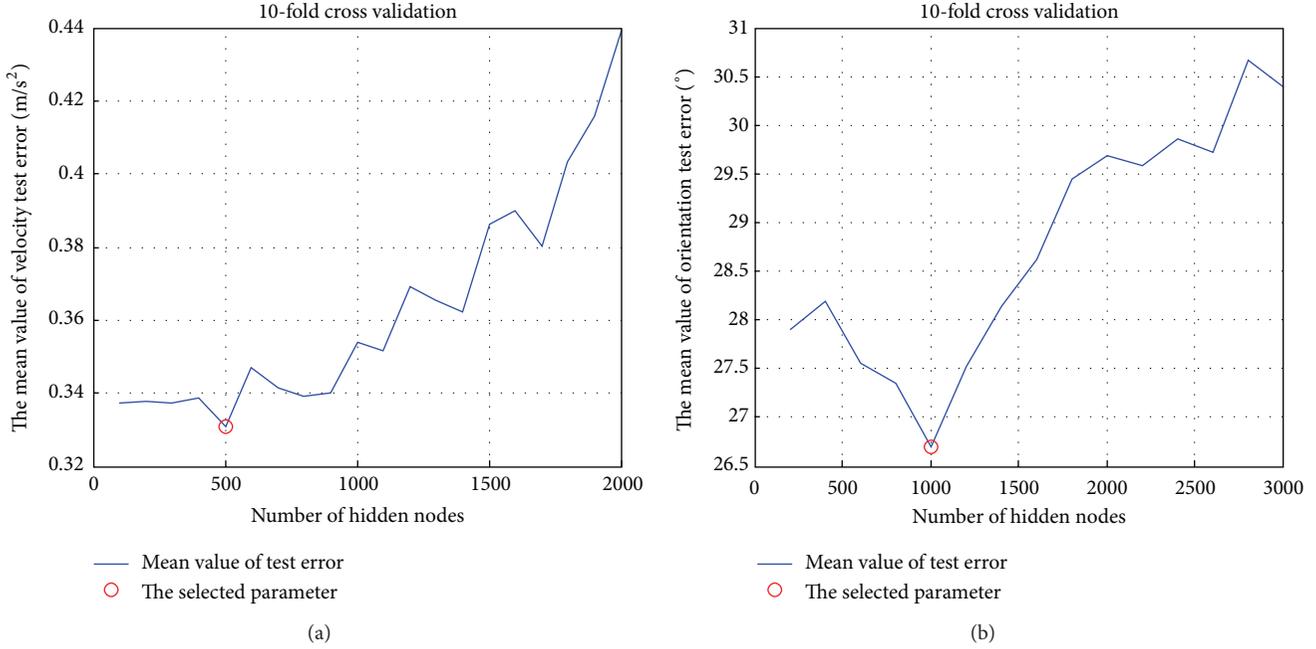


FIGURE 13: 10-fold cross validation to select number of hidden nodes (a) for velocity and (b) for orientation.

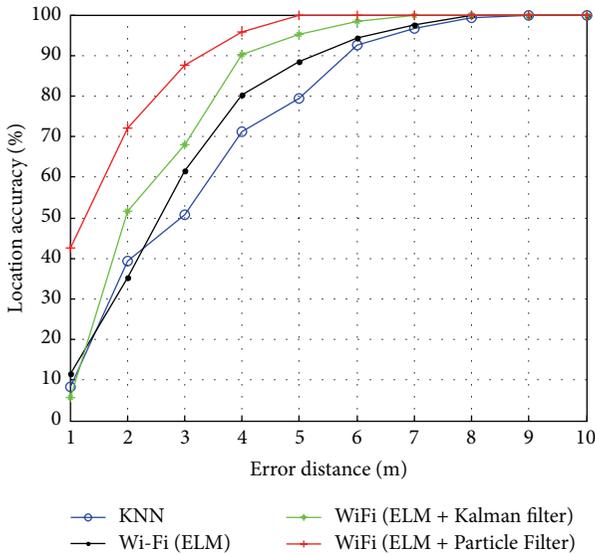


FIGURE 14: Location accuracy of four methods.

be seen in Figure 15(d). It is really better than those methods mentioned above. It can offer a timely and highly accurate location result.

8.3.2. *Experiments on Error Accumulation.* In order to value the error accumulation existing in sensors based location and how our method improved the accuracy with the help of Wi-Fi based location and positive impact of Particle Filter, we

calculate the distance error between the predicted position and the real one. And the result is shown in Figure 16. From that, we can conclude that when we use sensors only, the location error will accumulate into unbearable distance as time goes on. When we added Kalman filter to it, the location error increases more than before, which can be explained as that Kalman filter's function is to smooth the trajectory, not to minimize the location error. Fortunately, when we add Particle Filter, the location error decreases, but the error accumulation still leads the location accuracy divergence. When we use the Wi-Fi based location result to calibrate it within the Particle Filter framework, it will eliminate the error accumulation and keep the predicted position high precision.

8.3.3. *Evaluate the Location Time Consumption.* We also did the experiment of the time consumption of Wi-Fi based location and our methods.

From Figure 17 we can see that when we use Wi-Fi based location method on each kind of smart phone, it will take about 4.5 seconds. We can see the point indicating the location result jumping on the map. But using our method, the time consuming is almost 200 milliseconds, so that we can get a smoothly trajectory on the map.

## 9. Conclusion and Future Work

In this paper, we have presented a real-time and accurate indoor localization based on fusion model of Wi-Fi fingerprint and motion Particle Filter. The reason we raised this problem is that: a lot of sensors are integrated in smart phones, so we can easily get sensors data and Wi-Fi signals

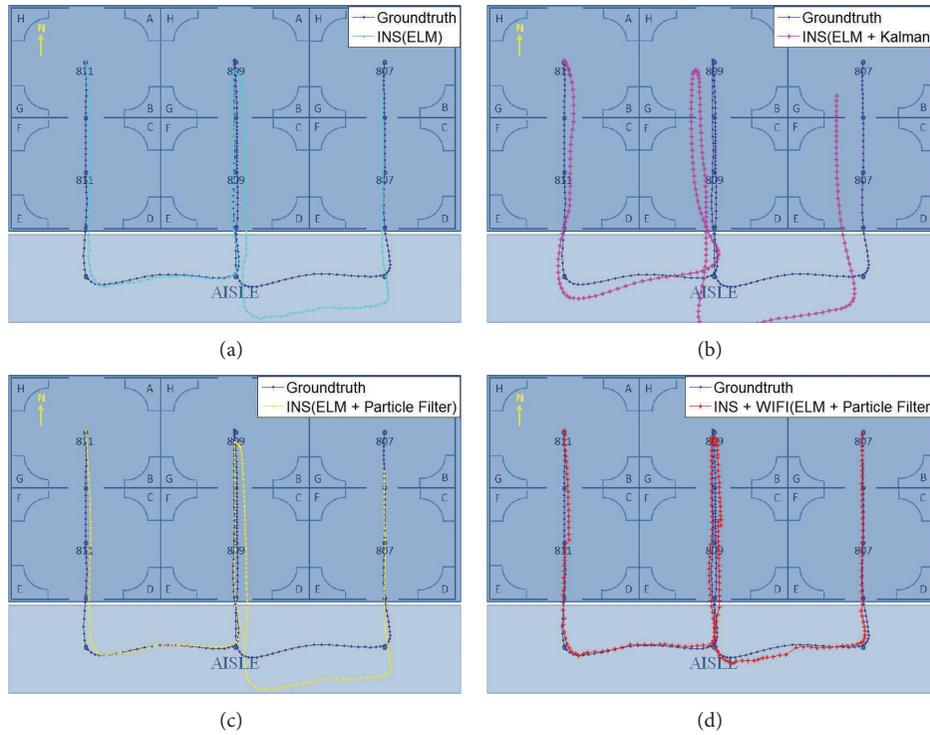


FIGURE 15: (a) Trajectory of sensors based location with ELM; (b) trajectory of sensors based location with ELM and Kalman filter; (c) trajectory of sensors based location with ELM and Particle Filter; (d) trajectory of our proposed method (INS: sensors data).

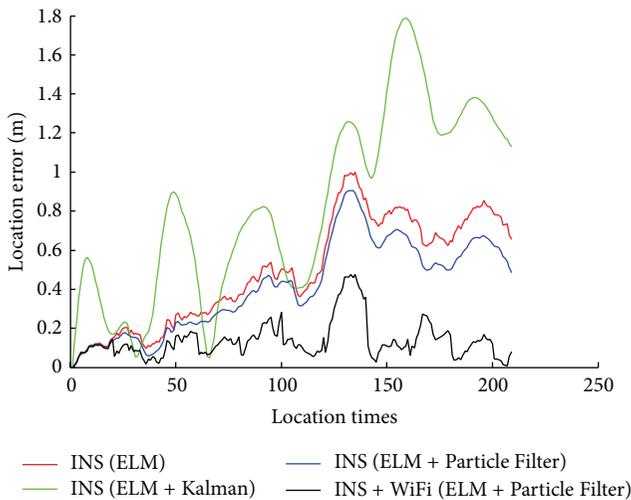


FIGURE 16: Location error of sensors based service and our proposed method (INS: sensors data).

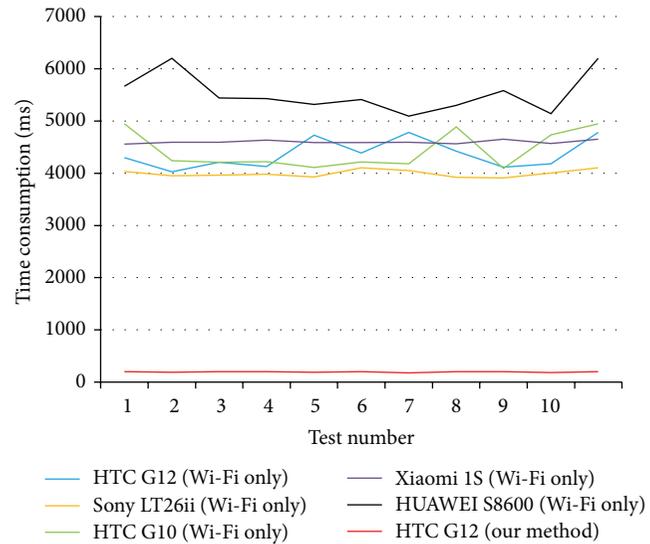


FIGURE 17: Time consumption of Wi-Fi based location and our method.

which can be used for indoor localization. But any single method cannot get satisfactory result. So we combine Wi-Fi based location and sensors based inertial navigation. And we also use Particle Filter and ELM algorithm to improve the location accuracy. From the experiments result, we can see that our method has achieved a good performance. But to get

a better accuracy, we think that another technology should be taken into account. RFID may be the suitable candidate. On the other hand, in order to speed up the localization service, maybe MapReduce framework [41] or Parallel Framework [42–46] can be involved.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is supported by NSFC General Program under Grant nos. 61173066 and 41201410 and Guangdong Province Funds Supported Project for the Development of Strategic Emerging Industry no. 2011912030.

## References

- [1] F. Evennou, F. Marx, and E. Novakov, "Map-aided indoor mobile positioning system using particle filter," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '05)*, vol. 4, pp. 2490–2494, March 2005.
- [2] Z. Chen, "Mining individual behavior pattern based on significant locations and spatial trajectories," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops '12)*, pp. 540–541, IEEE, March 2012.
- [3] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, pp. 775–784, IEEE, March 2000.
- [4] A. K. M. M. Hossain, H. Nguyen van, J. Yunye, and S. Wee-Seng, "Indoor localization using multiple wireless technologies," in *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS '07)*, pp. 1–8, Pisa, Italy, October 2007.
- [5] N. Swangmuang and P. Krishnamurthy, "An effective location fingerprint model for wireless indoor localization," *Pervasive and Mobile Computing*, vol. 4, no. 6, pp. 836–850, 2008.
- [6] Z. Sun, Y. Chen, J. Qi, and J. Liu, "Adaptive localization through transfer learning in indoor Wi-Fi environment," in *Proceedings of the 7th International Conference on Machine Learning and Applications (ICMLA '08)*, pp. 331–336, IEEE Computer Society, San Diego, Calif, USA, December 2008.
- [7] Y. Chen, J. Qi, Z. Sun, and Q. Ning, "Mining user goals for indoor location-based services with low energy and high QoS," *Computational Intelligence*, vol. 26, no. 3, pp. 318–336, 2010.
- [8] Z. Chen, S. Wang, Y. Chen, Z. Zhao, and M. Lin, "InferLoc: calibration free based location inference for temporal and spatial fine-granularity magnitude," in *Proceedings of the 15th IEEE International Conference on Computational Science and Engineering (CSE '12)*, pp. 453–460, Nicosia, Cyprus, December 2012.
- [9] Z. Chen, Y. Chen, S. Wang, and Z. Zhao, "A supervised learning based semantic location extraction method using mobile phone data," in *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE '12)*, pp. 548–551, Zhangjiajie, China, May 2012.
- [10] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, Artech House, 2nd edition, 2013.
- [11] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the USENIX Conference on USENIX Annual Technical Conference*, 2010.
- [12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, IEEE, July 2004.
- [13] J. Thomas and R. W. Levi, "Dead reckoning navigational system using accelerometer to measure foot impacts," U.S. Patent No. 5,583,776, 1996.
- [14] H. Leppäkoski, J. Käppi, J. Syrjärinne, and J. Takala, "Error analysis of step length estimation in pedestrian dead reckoning," in *Proceedings of the 15th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS '02)*, pp. 1136–1142, Portland, Ore, USA, September 2002.
- [15] "Motion Sensors," [http://developer.android.com/guide/topics/sensors/sensors\\_motion.html](http://developer.android.com/guide/topics/sensors/sensors_motion.html).
- [16] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A Reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 14th International Conference on Ubiquitous Computing*, pp. 421–430, ACM, September 2012.
- [17] I. J. Schoenberg, *Cardinal Spline Interpolation*, vol. 12, SIAM, Philadelphia, Pa, USA, 1973.
- [18] F. Evennou and F. Marx, "Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning," *EURASIP Journal on Applied Signal Processing*, vol. 2006, article 164, 2006.
- [19] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [20] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, June 2014.
- [21] J. Liu, Y. Chen, M. Liu, and Z. Zhao, "SELM: semi-supervised ELM with application in sparse calibrated location estimation," *Neurocomputing*, vol. 74, no. 16, pp. 2566–2572, 2011.
- [22] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.
- [23] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.
- [24] J. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [25] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [26] Y. Chen, Z. Zhao, S. Wang, and Z. Chen, "Extreme learning machine-based device displacement free activity recognition model," *Soft Computing*, vol. 16, no. 9, pp. 1617–1625, 2012.
- [27] Z. Chen, Y. Chen, L. Hu, S. Wang, and X. Jiang, "Leveraging two-stage weighted ELM for multimodal wearables based fall detection," in *Proceedings of ELM-2014 Volume 2: Applications*, vol. 4 of *Proceedings in Adaptation, Learning and Optimization*, pp. 161–168, Springer International Publishing, Cham, Switzerland, 2015.
- [28] Z. Chen, S. Wang, Z. Shen, Y. Chen, and Z. Zhao, "Online sequential ELM based transfer learning for transportation mode recognition," in *Proceedings of the 6th IEEE International Conference on Cybernetics and Intelligent Systems (CIS '13)*, pp. 78–83, IEEE, November 2013.

- [29] Z. Zhao, Z. Chen, Y. Chen, S. Wang, and H. Wang, "A class incremental extreme learning machine for activity recognition," *Cognitive Computation*, vol. 6, no. 3, pp. 423–431, 2014.
- [30] J.-F. Liu, Y. Gu, Y.-Q. Chen, and Y.-S. Cao, "Incremental localization in WLAN environment with timeliness management," *Chinese Journal of Computers*, vol. 36, no. 7, pp. 1448–1455, 2013.
- [31] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [32] K. Nummiaro, E. Koller-Meier, and L. van Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99–110, 2003.
- [33] F. Gustafsson, F. Gunnarsson, N. Bergman et al., "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.
- [34] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.
- [35] S. Thrun, "Simultaneous localization and mapping," in *Robotics and Cognitive Approaches to Spatial Mapping*, pp. 13–41, Springer, Berlin, Germany, 2008.
- [36] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom '12)*, pp. 269–280, ACM, August 2012.
- [37] M. Tüchler, A. C. Singer, and R. Koetter, "Minimum mean squared error equalization using a priori information," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 673–683, 2002.
- [38] X. Yin, J. A. N. Goudriaan, E. A. Lantinga, J. A. N. Vos, and H. J. Spiertz, "A flexible sigmoid function of determinate growth," *Annals of Botany*, vol. 91, no. 3, pp. 361–371, 2003.
- [39] T. L. Fine, *Feedforward Neural Network Methodology*, Springer, Berlin, Germany, 1999.
- [40] F. Evennou and F. Marx, "Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning," *Eurasip Journal on Applied Signal Processing*, vol. 2006, Article ID 86706, 2006.
- [41] T.-R. Hsiang, Y. Fu, C.-W. Chen, and S.-L. Chung, "A MapReduce-based indoor visual localization system using affine invariant features," *Computers & Electrical Engineering*, vol. 39, no. 7, pp. 2369–2378, 2013.
- [42] C. Yan, Y. Zhang, J. Xu et al., "Efficient parallel framework for HEVC motion estimation on many-core processor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, pp. 2077–2089, 2014.
- [43] C. Yan, Y. Zhang, J. Xu et al., "A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors," *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 573–576, 2014.
- [44] C. Yan, Y. Zhang, F. Dai, X. Wang, L. Li, and Q. Dai, "Parallel deblocking filter for HEVC on many-core processor," *Electronics Letters*, vol. 50, no. 5, pp. 367–368, 2014.
- [45] Y. Zhang, C. Yan, F. Dai, and Y. Ma, "Efficient parallel framework for H.264/AVC deblocking filter on many-core platform," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 510–524, 2012.
- [46] C. Yan, Y. Zhang, F. Dai, and L. Li, "Highly parallel framework for HEVC motion estimation on many-core platform," in *Proceedings of Data Compression Conference (DCC '13)*, pp. 63–72, March 2013.

## Research Article

# Research on Three-dimensional Motion History Image Model and Extreme Learning Machine for Human Body Movement Trajectory Recognition

**Zheng Chang, Xiaojuan Ban, Qing Shen, and Jing Guo**

*School of Computer and Communication Engineering, University of Science and Technology Beijing, Haidian District, Beijing 100083, China*

Correspondence should be addressed to Xiaojuan Ban; [banxj@ustb.edu.cn](mailto:banxj@ustb.edu.cn)

Received 15 August 2014; Revised 3 November 2014; Accepted 5 November 2014

Academic Editor: Zhan-li Sun

Copyright © 2015 Zheng Chang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Based on the traditional machine vision recognition technology and traditional artificial neural networks about body movement trajectory, this paper finds out the shortcomings of the traditional recognition technology. By combining the invariant moments of the three-dimensional motion history image (computed as the eigenvector of body movements) and the extreme learning machine (constructed as the classification artificial neural network of body movements), the paper applies the method to the machine vision of the body movement trajectory. In detail, the paper gives a detailed introduction about the algorithm and realization scheme of the body movement trajectory recognition based on the three-dimensional motion history image and the extreme learning machine. Finally, by comparing with the results of the recognition experiments, it attempts to verify that the method of body movement trajectory recognition technology based on the three-dimensional motion history image and extreme learning machine has a more accurate recognition rate and better robustness.

## 1. Introduction

With the rapid development of the natural human-computer interaction technology, the body movement trajectory tracking and recognition technology has become an important and indispensable research direction in the natural human-computer interaction technology. As we all know, since the body movement is a natural and intuitive communication mode [1], therefore, beyond all doubts, the body movement recognition technology has become a useful technology to the new generation of natural human-computer interaction interface [2–4], especially for the disabled and patients who can only use their body movements to give orders to the auxiliary equipment (such as the wheelchair, the smart television, and disabled scooter), which will bring them more convenience.

The prior body movement trajectory recognition researches on the human-computer interaction mainly focus on the modelling of human skin colour and the extraction of dynamic body movements based on image attributes

of the robust feature [5] and the artificial neural network; however, due to the diversity, ambiguity, and disparity in time and space of the body movements, the traditional body movement trajectory recognition researches have great limitations. The paper attempts to introduce the invariant moments of the three-dimensional motion history image and the extreme learning machine into the body movement trajectory recognition, which will make the machine vision recognition of the body movement trajectory more accurate, efficient, and robust [6].

The paper is organized as follows. Section 1 describes the background of the human body movement trajectory recognition and the importance of the human body movement trajectory recognition. Section 2 describes the main problems in the human body movement trajectory recognition, followed by a summary of the normal algorithm. Section 3 describes the principal ideas of our new method and the main steps including the feature extraction of the motion history image, calculation of the invariant moments, and the movement recognition based on extreme learning machine. Section 4



FIGURE 1: Two-dimensional body movement image.

makes a comparison of the recognition performance between our new method and the other algorithms. Section 5 summarizes the conclusions of this study.

## 2. Problem Description

The essence of the body movement trajectory recognition is to extract motion features and to classify the sample data accurately. It is a contrast between the body movement trajectory captured by the sensor and the predefined sample movement trajectory. Therefore, there are two steps in the body movement trajectory recognition. The first step is the features extraction of dynamic body movements. And the second step is the classification of the body movement trajectory captured by the sensor.

In the first extraction step, the traditional method extracts the dynamic body movements by applying the hidden Markov model [7, 8] as is shown in Figure 1.

In the hidden Markov model method, human movement trajectory data  $\mathcal{M}$  can be regarded as the state series  $\mathcal{M} = (\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \dots, \mathcal{F}_T)$  which is obtained from sampling frames. In one single frame, the state of the human movement is represented by  $\mathcal{F} = (r_1, r_2, r_3, \dots, r_n)$ , wherein  $r_n$  represents the value of the  $n$ th characteristic values in the current frame. The movement trajectory recognition is the process of comparing the real-time trajectory captured by sensor with the state series  $\mathcal{M}$  and then matching  $\mathcal{M}$  with the state series of the predefined sample trajectory.

Based on the hidden Markov model (HMM), the whole process of the comparison between the real trajectory from the sensor and the predefined sample is shown in Figure 2 [9]. But based on the hidden Markov model (HMM), the recognition process still has some limitations; for example, consider the following.

- (i) Light: when the light condition is changed, the luminance information of the body will change since the images captured by the sensor are easily affected by natural light and artificial light. Different skin colors can also easily affect the luminance information under the same light condition.
- (ii) Obstruction: during the whole process, the body movement trajectory may be blocked by some objects in the environment or the other parts of the body since the obstruction can lead to the loss of the identification information of body, which will affect the reliability of the body movement recognition greatly.
- (iii) Background: in the real-time body movement recognition process, if the factors (color, texture, shape,

etc.) of the body movement area and the background area are similar, it will also affect the performance of the recognition [10].

The three-dimensional hidden Markov model (3DHMM) is a better performance, but this method has been restricted to very few application fields because of the huge amount of calculation, the inefficiency of the training, and the easy accessibility to the local optimal value, and so forth.

In the second classification step, the traditional methods involve many machine learning algorithms (such as  $K$  nearest neighbor method and gradient descent-based feed-forward network learning methods) [11–13]. But these methods still have some limitations as follows.

- (i)  $K$  nearest neighbor method: the  $K$  nearest neighbor method (KNN) needs to compute the distance (such as Euclidean distance, Mahalanobis distance, or Pearson correlation) between the real body movement trajectory captured by the sensor and every predefined sample body movement trajectory. Therefore, because of the heavy computation in this method, it has been restricted to very few application fields.
- (ii) Gradient descent-based feed-forward network learning methods: the BP artificial network is one of the typical gradient descent-based feed-forward network learning methods. All the parameters of the feed-forward networks need to be turned and thus there exists the dependency between different layers of parameters (weight and biases) in the BP artificial network. Therefore, the training process is very slow. And this gradient descent-based learning method may easily converge to local minima and overfitting problem [14–16].

To solve these problems, this paper attempts to combine the three-dimensional motion history image and the extreme learning machine in order to overcome those shortcomings.

In the comparative experiment section (Section 4.3), a detailed and specific experiment about the comparison of the different classification methods will be conducted.

The three-dimensional motion history image can easily present the feature of the human body movement trajectory, such as the space feature and the time feature. And the seven invariant moments of the three-dimensional motion history image have translation invariance, scaling invariance, and rotation invariance. These invariances could tactfully overcome the observation position sensitivity of the human body movement trajectory.

The comparison between the motion history image and the traditional MHH method will be described in the comparative experiment at the end of Section 4.2.

The extreme learning machine has a faster and better generalization performance than another traditional feed-forward artificial network (such as BP network) [17–19]. What is more, the extreme learning machine could obtain the global optimal value easily. Therefore, it adopts a new method to achieve the body movement trajectory recognition goal in this paper (Figure 3).

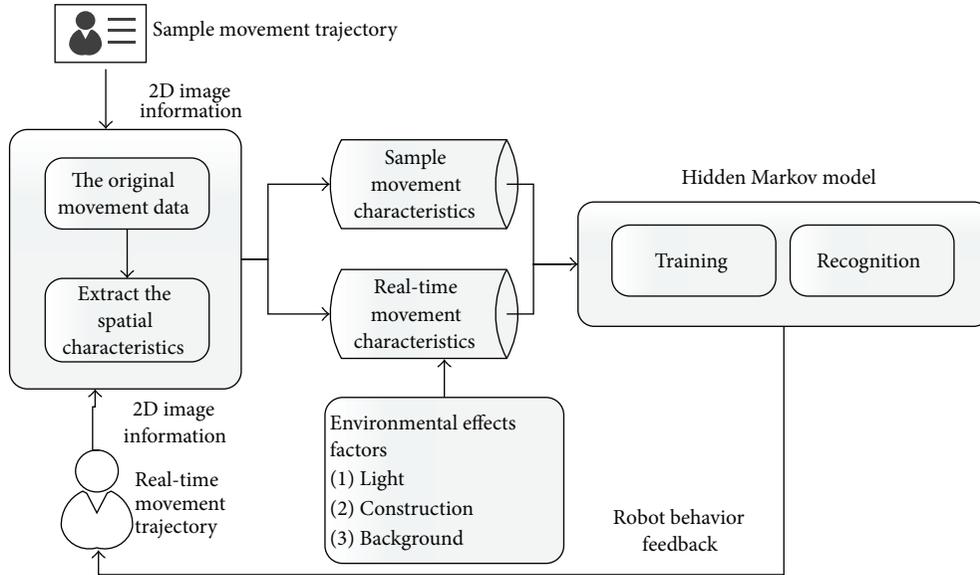


FIGURE 2: The HMM based on two-dimensional image.

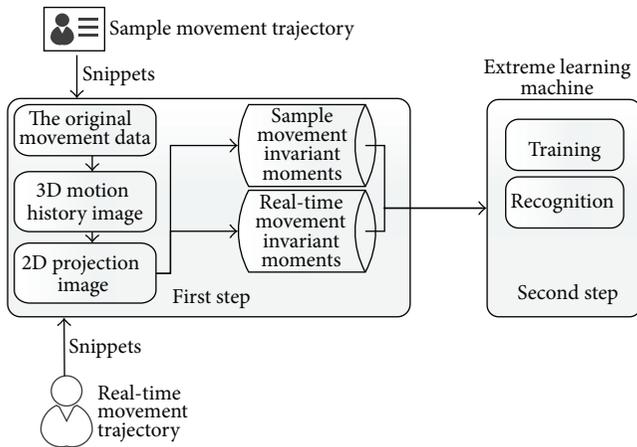


FIGURE 3: The ELM based on 3D motion history image.

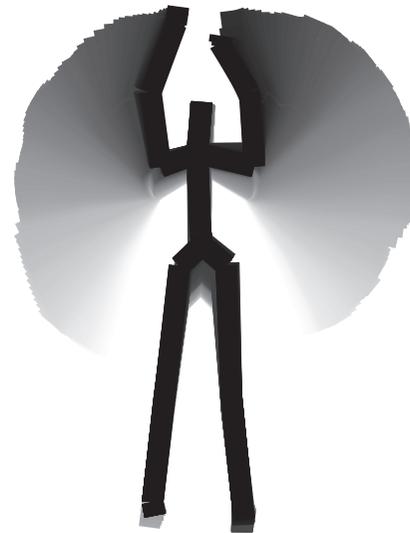


FIGURE 4: The MHI based on three-dimensional depth data.

In the first step, this paper attempts to combine the motion history image (MHI) with the three-dimensional depth data of the body movements in order to get the three-dimensional motion history image (3DMHI) of body movements (Figure 4) because a single human body gesture cannot show the meaning of the human action. Using this method we could get the human body movement trajectory presentation from a few of previous video frames or snippets. The snippets which we set in our experiments usually take three minutes or more. In order to improve the accuracy of the human body movement trajectory recognition process, it adopts a high sampling rate, thirty frames per second in the paper.

At the end of Section 3.1, the grey scale of the motion history image (MHI) will be given a detailed and popular description.

And then it calculates seven invariant moments of the three-dimensional motion history image working as the eigenvector of the body movements. To calculate the invariant moments, after getting the three-dimensional motion history image, the image was projected in the XY plane, YZ plane, and XZ plane. In each projection plane, a set of invariant moments is calculated. Each three sets of invariant moments work as the eigenvector of the human body movement trajectory.

In the second step, this paper attempts to use the extreme learning machine instead of the traditional methods (like KNN, BP, or SVM). On the one hand, not only is the new process free from the effects of the illumination, obstruction,

background, and other environmental factors but it also improves the efficiency, accuracy, and robustness of body movement recognition. The extreme learning machine has a faster and better generalization performance than another traditional feed-forward artificial network (such as BP network) [20–22]. According to the previous splendid work, the extreme learning machine tends to have a better scalability and achieve a similar (for regression and binary class cases) and much better generalization performance (for multiclass cases) at a much faster learning speed (up to thousands of times) than traditional machine learning methods (such as KNN, BP, and SVM) [23–25].

### 3. Problem Solving

**3.1. Body Movements Characterized by 3D Motion History Image.** To characterize the 3D motion information, the paper proposes a new method named three-dimensional motion history images approach. This method improves the application of the traditional motion history images approach based on two-dimensional image in order to combine with the three-dimensional depth data. By combining the invariant moments of the three-dimensional motion history image (computed as the eigenvector of body movements) and the extreme learning machine (constructed as the classification artificial neural network of body movements), the paper applies the method to the machine vision of the body movement trajectory.

The motion history image approach is a kind of special finite-difference time-domain method; it is a branch of the Finite Difference Time Domain (FDTD) method [26, 27]. The mechanism of the FDTD method is to get different images from continuous image sequences by comparing with two or three adjacent pixels in the corresponding frames and then to extract the human body moving regions in the image by setting the threshold. By introducing the 3D data, the paper presents the improved FDTD method named three-dimensional motion history image approach as follows:

$$\mathcal{D}(x, y, z, n) = \mathcal{F}(x, y, z, n-1) - 2\mathcal{F}(x, y, z, n) + \mathcal{F}(x, y, z, n+1). \quad (1)$$

Among them,  $N$  means the number of the current frame and  $N$  mainly shows the relation between the adjacent frames.  $\mathcal{F}(x, y, z, n)$  represents the pixel grey value in the position  $(x, y, z)$  in three-dimensional space;  $\mathcal{D}(x, y, z, n)$  is the result of the three consecutive frames' difference and also represents body movement changed area. The threshold  $\mathcal{D}(x, y, z, n)$  is as follows:

$$\mathcal{B}(x, y, z, n) = \begin{cases} 1 & \mathcal{D}(x, y, z, n) > \Gamma \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where  $\Gamma$  is the specially selected threshold. If the value is too low, it cannot effectively remove noises in images, but if the value is too high, it will impede the valuable variation of the image. So the value of the threshold should be adjustable for different experiment conditions. The experiment should be

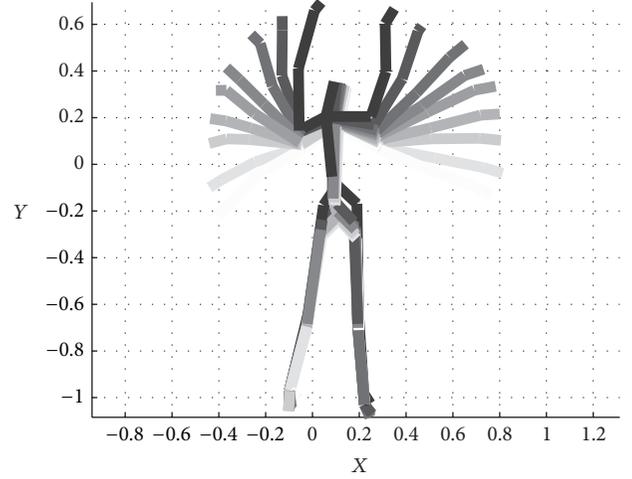


FIGURE 5: Three-dimensional motion history image of body movements (MHI).

repeated many times to determine the value of the threshold. Under my experiment condition, the value of the threshold is  $(10, 10, 15, n)$  in my follow-up work.

Three-dimensional motion history image approach of body movements is as follows:

$$\begin{aligned} \mathcal{H}_\tau(x, y, z, t) &= \begin{cases} \tau & \mathcal{B}(x, y, z, t) = 1 \\ \max(0, \mathcal{H}_\tau(x, y, z, t-1) - 1) & \text{otherwise.} \end{cases} \end{aligned} \quad (3)$$

Among them,  $\mathcal{H}_\tau(x, y, z, t)$  represents the pixel gray value in the position  $(x, y, z)$  and  $t$  in three-dimensional motion history image. The motion history image MHI not only reflects the external shape of the body movements (space feature) but also reflects the direction and state of them (time feature).

Generally speaking, a sequence of the human body movement trajectory image is compressed into a single and special image by the previous algorithm. The single and special image works as the motion history image (MHI).

In the motion history image, the grey value of each pixel is in proportion with the duration of the body movement in the position. The recent body gestures have the maximum grey value. Grey value changes reflect the direction of the body movements (Figure 5). With time elapsing, the grey value of the human body movement trajectory would decrease.

In other words, a sequence of the human body movement trajectory image is compressed into a single and special image by the previous algorithm. But each image's gray scale is changed. The longer the motion lasts in the sequence of the human body movement trajectory image, the lighter the gray scale presents.

**3.2. The Calculation of the Invariant Moments of the Motion History Image.** Although the three-dimensional motion history image approach based on the MHI is simple and efficient,



FIGURE 6: XY surface projection of the MHI.



FIGURE 8: XZ surface projection of the MHI.



FIGURE 7: YZ surface projection of the MHI.

it is too sensitive to the observation position. Seen from different observation positions, the three-dimensional motion history image will have different results. These differences will affect the accuracy of human body movement recognition process greatly. In order to overcome this shortcoming, this paper selects the invariant moments as eigenvector of the motion history image. The method of invariant moments is a classical method to extract image feature. Its translation invariance, scaling invariance, and rotation invariance properties rule out the impact on the position, distance, and angle.

To calculate the invariant moments, after getting the three-dimensional motion history image, our method attempts to project it in the XY plane (Figure 6), YZ plane (Figure 7), and XZ plane (Figure 8). This projection could simplify the invariant moment's calculation process. The

direct calculation of the three-dimensional motion history image could bring huge amounts of three-dimensional invariant moment's calculation instead of a much better performance of the human body movement trajectory recognition process. This method can get three views of three-dimensional motion history image with one motion. Then the calculation of invariant moments for the three main views is obtained.

For a size of  $\mathcal{M} \times \mathcal{N}$  digital image  $f(x, y)$ , the  $p + q$  order moment  $m_{p,q}$  is defined as follows:

$$m_{p,q} = \sum_{x=1}^N \sum_{y=1}^M f(x, y) x^p y^q. \quad (4)$$

Among them,  $p, q = 0, 1, 2, \dots$

$p + q$  order central moment  $\mu_{p,q}$  is defined as follows:

$$\mu_{p,q} = \sum_{x=1}^N \sum_{y=1}^M f(x, y) (x - \bar{x})^p (y - \bar{y})^q, \quad (5)$$

where  $(x, y)$  represents the object image point and  $(\bar{x}, \bar{y})$  is the object centroid:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}. \quad (6)$$

Among them,

$$\begin{aligned} m_{00} &= \sum_{x=1}^N \sum_{y=1}^M f(x, y), \\ m_{10} &= \sum_{x=1}^N \sum_{y=1}^M f(x, y) x^1, \\ m_{01} &= \sum_{x=1}^N \sum_{y=1}^M f(x, y) y^1. \end{aligned} \quad (7)$$

Then through the normalizing of the central moment by the zero-order central moments  $\mu_{00}$ , the normalized center moment of the motion history image could be got:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^r}, \quad r = \frac{p+q+2}{2}, \quad p+q = 2, 3, 4, \dots \quad (8)$$

Ming-kuei Hu got seven invariant moments based on the linear combination of two-order and three-order normalized central moment. And the seven invariant moments of the three-dimensional motion history image have translation invariance, scaling invariance, and rotation invariance. These invariances could skillfully overcome the observation position sensitivity of the human body movement trajectory. The image translation, rotation, and scaling are unchanged and the invariant moments are as follows [28]:

$$\begin{aligned} M_1 &= \eta_{20} + \eta_{02}, \\ M_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2, \\ M_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2, \\ M_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2, \\ M_5 &= (\eta_{30} - 3\eta_{12}) \times (\eta_{30} + \eta_{12}) \\ &\quad \times [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad + (\eta_{03} - 3\eta_{21})(\eta_{03} + \eta_{21}) \\ &\quad \times [(\eta_{03} + \eta_{21})^2 - 3(\eta_{12} + \eta_{30})^2], \\ M_6 &= (\eta_{20} - \eta_{02}) [(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ &\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}), \\ M_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{21}) \\ &\quad \times [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad - (3\eta_{12} - \eta_{30})(\eta_{03} + \eta_{21}) \\ &\quad \times [(\eta_{03} + \eta_{21})^2 - 3(\eta_{12} + \eta_{30})^2]. \end{aligned} \quad (9)$$

Because the values of the invariant moments are too small, it is compressed by the absolute value of the logarithm and so the actual values need to be amended in accordance with the following formula:

$$M_k = \log |M_k|, \quad k = 1, 2, 3, 4, 5, 6, 7. \quad (10)$$

Because formula (10) does not change the feature of the invariant moments, the seven new values of the invariant moments of the three-dimensional motion history image have translation invariance, scaling invariance, and rotation invariance. The invariant moments still have translation, rotation, and scaling invariance after amendment.

Through the calculation of the projection images in three directions, we will get a  $3 \times 7$  eigenvalue matrix. This eigenvalue matrix is the eigenvector for motion history image

because the three-dimensional motion history image could easily present the human body movement trajectory including the space feature and the time feature. And the seven invariant moments of the three-dimensional motion history image have translation invariance, scaling invariance, and rotation invariance. These  $3 \times 7$  eigenvalue matrixes could tactfully overcome the observation position sensitivity of the human body movement trajectory and present the human body movement trajectory easily.

*3.3. The Body Movements Recognition Based on Extreme Learning Machine.* In the process of recognition, first the samples of body movement are collected and then a training sample set was built to obtain a better recognition performance. In order to get a better human body movement trajectory recognition performance, the samples of the human body movement trajectory must be definite, clear, and slow; in other words, all of the training samples must comply with the criterion under my experiment conditions.

For the same body movement, different people involved should repeat the action several times. And then multiple groups of three-dimensional motion history images are collected for each human body movement trajectory; after that, the training sample set for each human body movement trajectory will be established.

According to many precedent outstanding researches, we can draw the conclusion that the input weights and hidden layer biases of a single-hidden layer feed-forward neural network (SLFN) can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable [29].

Based on the conclusion, this paper proposes the extreme learning machine (ELM) for SLFN. The extreme learning machine need not adjust the most parameters of the artificial neural network repeatedly, such as the input weights and the hidden layer biases. Contrary to the traditional artificial neural network, this randomly assigned approach could reduce huge amounts of the calculation and increase the artificial neural network's training efficiency. What is more, the precedent researcher has proved that this simplified approach does not reduce the test accuracy of the artificial neural network. Because of these advantages, the ELM could transform the complex machine learning problems into a simple linear problem which could be determined through a generalized inverse calculation of the hidden layer weight matrices.

In some applications, the extreme learning machine tends to have a better scalability and achieve a similar (for regression and binary class cases) and much better generalization performance (for multiclass cases) at a much faster learning speed (up to thousands of times) than traditional machine learning methods (such as KNN, BP, and SVM).

Because the ELM can have much better efficiency and accuracy; moreover, it can overcome the traditional methods' shortcomings. For our body movement trajectory recognition process, the paper attempts to construct a SLFN (Figure 9) as follows.

There are  $\mathcal{N}$  hidden nodes and  $\mathcal{M}$  input lay nodes in our SLFN. Our training sample set is  $(X_j T_j)$ , where

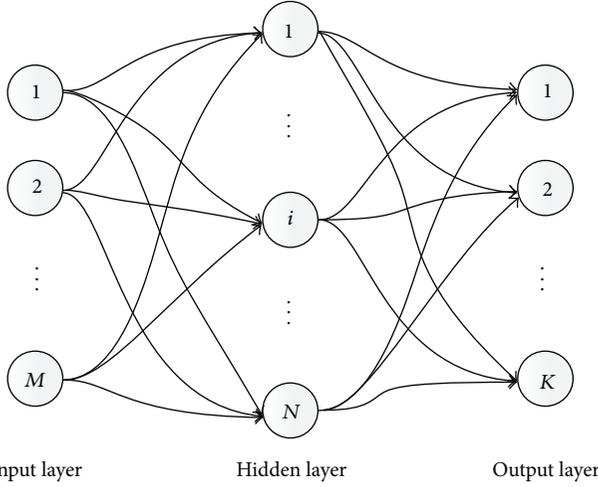


FIGURE 9: ELM artificial neural network.

$X_j = [X_{j1}, X_{j2}, \dots, X_{jM}]$  is the invariant moments of training samples and  $T_j = [T_{j1}, T_{j2}, \dots, T_{jM}]^T$  is the kind of training samples. There are  $K$  output layer nodes; it means that there are  $K$  types of human body movement trajectory. Each node in the output layer presents a type of human body movement trajectory. And activation function  $f(x)$  is mathematically modelled as

$$\sum_{i=1}^{N \cdot M} \beta_i f(W_i \cdot X_j + b_i) = R_j, \quad (11)$$

where  $R_j = [R_{j1}, R_{j2}, \dots, R_{jM}]$  is the output vector from our SLFN,  $W_i = [W_{i1}, W_{i2}, \dots, W_{iN}]^T$  is the weight vector which connects the  $i$ th hidden layer node and the input nodes,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{iN}]^T$  is the weight vector which connects the  $i$ th hidden layer node and the output nodes, and  $b_i$  is the threshold of the  $i$ th hidden layer node.

Each element in the two vectors  $T_j = [T_{j1}, T_{j2}, \dots, T_{jM}]^T$  and  $R_j = [R_{j1}, R_{j2}, \dots, R_{jM}]$  is a  $K$  dimensional vector. In other words,  $T_{ji}$  and  $R_{ji}$  serve as the real type and computational type of the invariant moments of training sample  $X_{ji}$ , respectively.

Our goal is to make  $T_j$  close to  $R_j$ , which is equivalent to minimizing the cost function:

$$C = \sum_{i=1}^{N \cdot M} \beta_i f(W_i \cdot X_j + b_i) - T_j = R_j - T_j. \quad (12)$$

Previous excellent papers have strict calculation that SLFN with  $N$  hidden nodes can distinguish  $N$  samples exactly for any infinitely differentiable activation function and SLFN may require less than  $N$  hidden nodes if learning error is allowed [30]. Of course, the number of the hidden layer nodes in the artificial neural network could have some impacts on the human body movement trajectory recognition performance. In the experiment section, this paper will show these impacts under different numbers of the hidden layer nodes.

The equation is presented as follows:

$$\sum_{i=1}^{N \cdot M} \beta_i f(W_i \cdot X_j + b_i) = T_j. \quad (13)$$

The above equations could be abbreviated as

$$BF = T, \quad (14)$$

where

$$F = \begin{bmatrix} f(W_1 \cdot X_1 + b_1) & \cdots & f(W_1 \cdot X_M + b_1) \\ \vdots & \cdots & \vdots \\ f(W_N \cdot X_1 + b_N) & \cdots & f(W_N \cdot X_M + b_N) \end{bmatrix}_{N \times M},$$

$$B = [\beta_1, \beta_2, \dots, \beta_N],$$

$$T = [T_1, T_2, \dots, T_M]. \quad (15)$$

On both sides of the equation, the transpose operation could be applied in order to obtain the standard ELM model:

$$\mathcal{H}\beta = \mathcal{T}, \quad (16)$$

where

$$\mathcal{H} = F^T,$$

$$\beta = B^T, \quad (17)$$

$$\mathcal{T} = T^T.$$

So we could obtain the result like this:

$$\beta = \mathcal{H}^\dagger \mathcal{T}, \quad (18)$$

where  $\mathcal{H}^\dagger$  is the Moore-Penrose generalized inverse of matrix  $\mathcal{H}$ .

If  $N = M$  and  $\mathcal{H}$  has an inverse, the solution is existing and unique. And the answer is, evidently,  $\mathcal{H}^\dagger = \mathcal{H}^{-1}$ . The result is shown as follows:

$$\beta = \mathcal{H}^{-1} \mathcal{T}. \quad (19)$$

There are several methods which could be used to calculate the Moore-Penrose generalized inverse of matrix  $\mathcal{H}$ . This paper attempts to apply the spectral theorem and Tikhonov's regularization method to calculate  $\mathcal{H}^\dagger$  Moore-Penrose generalized inverse of matrix  $\mathcal{H}$  [31]:

$$\mathcal{H}^\dagger = \sum_{\substack{b=1 \\ \alpha_b \neq 0}}^s \frac{1}{\alpha_b} \left[ \prod_{\substack{l=1 \\ l \neq b}}^s (\alpha_b - \alpha_l)^{-1} \right]$$

$$\times \left[ \prod_{\substack{l=1 \\ l \neq b}}^s (\mathcal{H}^* \mathcal{H} - \alpha_l E_n) \right] \mathcal{H}^*, \quad (20)$$

where  $\mathcal{H}^*$  represents the adjoint matrix of  $\mathcal{H}$  and  $\alpha_b, b = 1, 2, \dots, s$ , are the eigenvalues of  $\mathcal{H}^* \mathcal{H}$  (or the singular values of  $\mathcal{H}$ ).

According to the training sample set, we could obtain the weight vector  $\beta$  and then complete the training of ELM. As presented in the previous outstanding paper, ELM has many important properties, as follows:

- (i) the smallest norm of weights;
- (ii) minimum approximation error;
- (iii) the minimum norm least-squares solution of  $\mathcal{H}\beta = \mathcal{T}$  which is unique.

After the training process, the invariant moments of real-time human body movement trajectory captured by the structured light sensor can be put into the ELM. In the process of the pretrained ELM, the type of the real-time human body movement trajectory could be obtained. Although the training process of the ELM would cost some time, the recognition process of the real-time human body movement trajectory is very fast. In the following experiment analysis section, this paper will present the accuracy and efficiency of this method.

#### 4. Results of the Experiment

Supported by the laboratory's National Science Funding, the researches on the human body movement trajectory recognition are required to be conducted under the office environment, and the experimental data in this paper therefore are required to be specific and unique. The public data sets are more universal and unsuitable for our specific research field. In consequence, the human body movement trajectory data set needs to be collected by ourselves.

In the experiment of different lighting conditions, four people are asked to do four kinds of body movements, as is shown in Figures 10, 11, 12, and 13. Each kind of body movements is repeated 10 times and generates 40 samples for each body movement. Every movement lasts five to fifteen seconds with the image size of  $1200 \times 900$ .

In the other experiments, if we adopt same raw data in the first experiment, the testing process will be too quick to record the testing time. So the other five people whose heights are from 155 cm to 180 cm are tested. The five people are asked to do five kinds of body movements. These movements and their projection images are shown in Figures 17, 18, 19, 20, and 21.

All the data preprocessing and the comparative experiments (including Sections 4.2, 4.3, 4.4, 4.5, and 4.6) are carried out in MATLAB 2010b environment running in Intel Core 4 Quad 2.9 GHZ CPU with 6 GB RAM. Microsoft's structured light sensor Kinect serves as the three-dimension human body movement trajectory data capture sensor to capture the human body movement trajectory.

*4.1. Experiment Condition and Data Preprocessing.* This human body movement trajectory recognition experiments are done in normal laboratory environment. In the experiment, people should keep the body forward, perpendicular

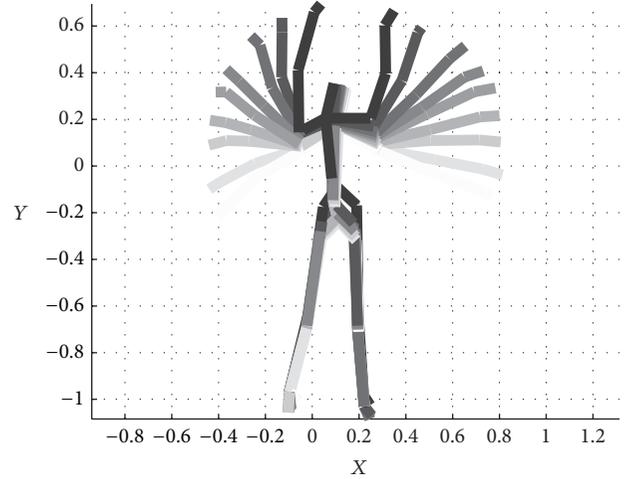


FIGURE 10: Motion history image for movement A.

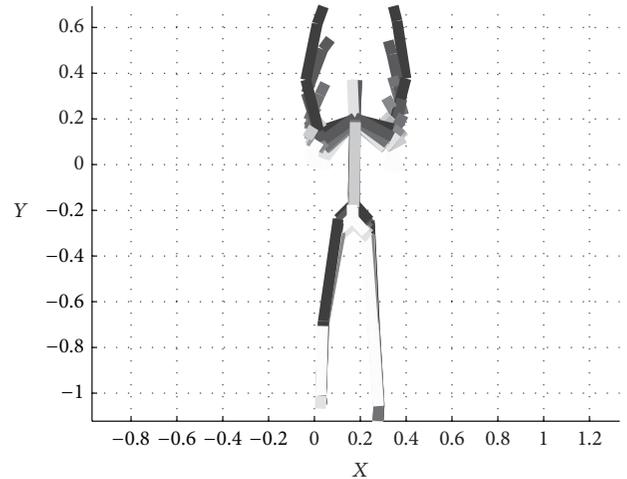


FIGURE 11: Motion history image for movement B.

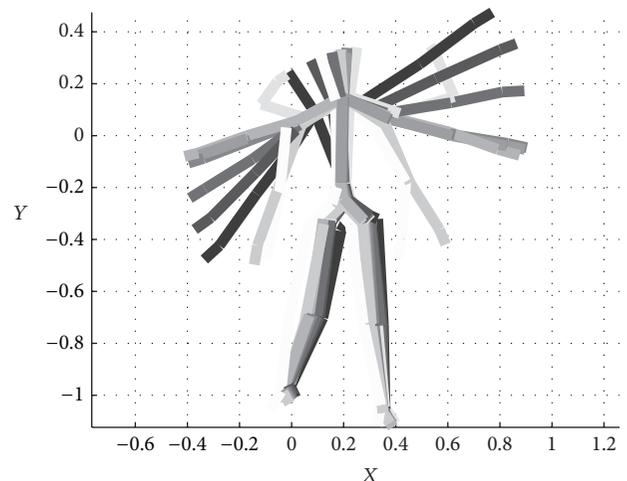


FIGURE 12: Motion history image for movement C.

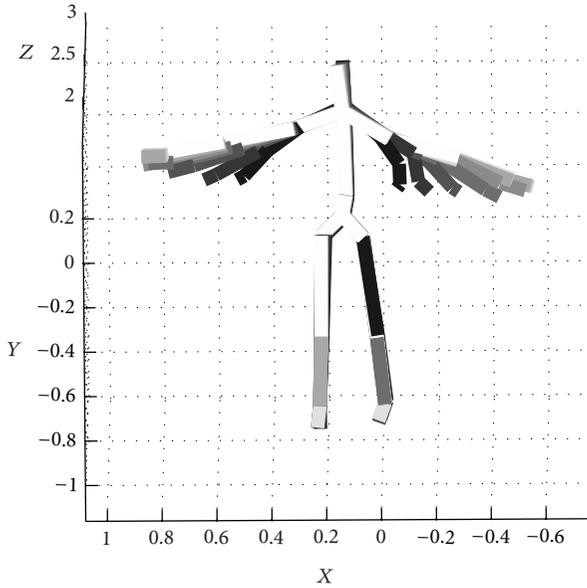


FIGURE 13: Motion history image for movement D.

to the horizontal plane, and be about 1.2 to 2 meters to the structured light sensor.

As is mentioned previously, the human body movement trajectory of a single motion has been collected. The motion history image is constructed with whole image sequences as in Figure 1. In consequence, there is no need to find the key frame and detect human motion in some motion image sequences. Each frame in the motion image sequences is equal and important for the three-dimension motion history image. The three-dimension motion history image from the whole motion image sequences is calculated. Besides the three-dimension motion history image represents the feature of human motion.

In this paper, the physical movements monitored are debounced and the centre position data of the prior frame are recorded to compare with the centre position data of the current frame. If the deviation is within the threshold range, the position data of the prior frame is chosen to neglect the jitter of the current frame [27, 32].

During the comparative experiments, repeated experiments are conducted many times to determine the value of the threshold. In the end, the paper sets the value of the threshold as 100 pixels in the follow-up work.

When using the real-time human body movement trajectory, invalid frames will appear at the beginning and the end of the movement. By eliminating the jitter of the physical movements, we could remove the useless part of the physical movements, and all the frames left are presented clearly.

As is mentioned previously, MATLAB program is used to preprocess all raw human body movement trajectory data. Then the motion history image with the preprocessed human body movement trajectory data is calculated.

4.2. The Experiments under Different Feature Extracting Algorithms. In order to verify the robustness of the MHI method

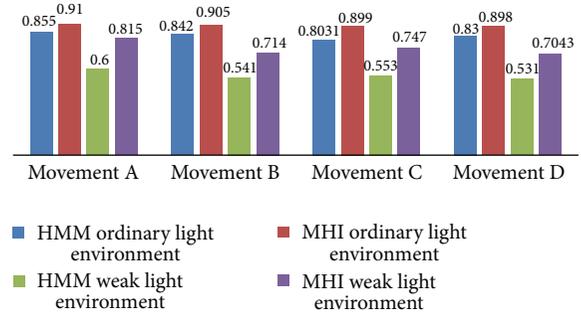


FIGURE 14: Recognition rate in different lighting conditions.

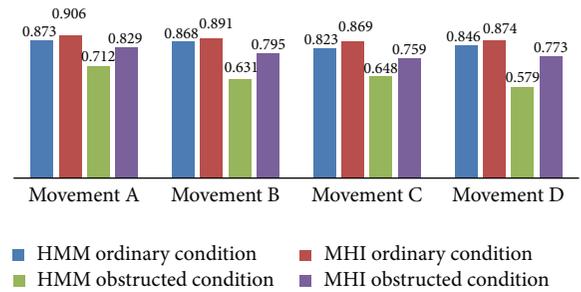


FIGURE 15: Recognition rate in different obstruction conditions.

in the extracting feature step, here some comparing experiments are carried out under different conditions, such as different light, different obstruction, and different background factor. The recognition rate between the HMM method and the MHI method is under different conditions checked.

In the experiment, four people are asked to do four kinds of body movements, as is shown in Figures 10, 11, 12, and 13. Each kind of body movements is repeated 10 times in the different conditions. It generates 40 samples for each body movement in each experiment condition. Every movement lasts five to fifteen seconds with the image size of 1200 × 900.

In order to keep the same experiment condition, the same human body movement trajectory raw data is adopted. The first 20 samples of each kind of movements are chosen as training data to get the standard movement templates using the traditional hidden Markov model (HMM) and three-dimensional motion history image approach (MHI) in each experimental condition. Then the other 20 samples left are used as testing data in each experimental condition.

Figure 14 is the recognition accuracy in normal light and the weak light for every human body movement trajectory. The recognition accuracy rate declines sharply by the traditional method in low light conditions. The three-dimensional motion history image approach with 3D depth data could capture the trajectory of human body movement very well even under weak light environment. The experiment shows that the new method has better robustness under the weak light conditions than the traditional method.

By adopting the hat, glass, or mask as obstruction, Figure 15 is the recognition accuracy in different obstruction experimental condition for each kind of human body

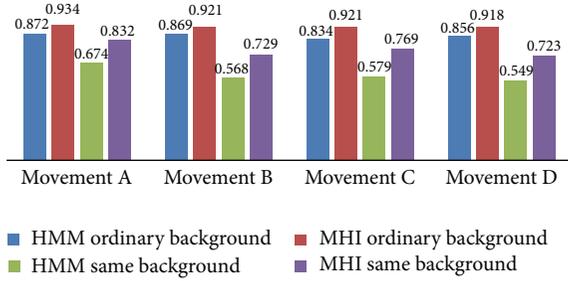


FIGURE 16: Recognition rate in different background conditions.

movement trajectory. The recognition accuracy rate declines sharply by the traditional method in strange obstruction experimental condition, such as hat, glass, or mask. The three-dimensional motion history image approach with 3D depth data will capture the trajectory of human body movement very well even under obstruction experimental condition. The experiment shows that the new method has a better robustness under the obstruction experimental conditions than the traditional method.

By adopting some texture and pattern wall as comparing background, Figure 16 is the recognition accuracy in the different background conditions for each human body movement trajectory. As we can see, the recognition accuracy rate declines sharply by the traditional method in same texture and pattern background. The three-dimensional motion history image approach with three-dimensional depth data will capture the trajectory of human body movement very well even under the same texture and pattern background condition. The experiment shows that the new method has better robustness under the same texture and pattern background conditions than the traditional method.

In the favourable conditions, the recognition rate of the MHI method and the HMM method is almost the same, but in the unfavourable conditions (weak light, obstruction, and same background) the MHI method has a higher recognition rate than the traditional HMM method.

According to the result of the comparing experiments, the conclusion could be safely drawn that the MHI method has a better performance than HMM method.

#### 4.3. The Experiments under Different Recognition Algorithms.

In order to verify the efficiency and accuracy of the body movement recognition process based on the extreme learning machine, here some experiments under different recognition algorithms are carried out (ELM, KNN, BP, and SVM).

We could compare with the recognition rate and training time of different recognition algorithms (ELM, KNN, BP, and SVM) in the same raw data and experimental conditions.

In these contrast experiments, if we adopt the same raw data in the first experiment, the testing process will be too quick to record the testing time. Therefore more human body movement trajectories are collected and other five people whose heights are from 155 cm to 180 cm are tested. Five people are asked to do five kinds of body movements. These movements and their projection images are as shown in Figures 17, 18, 19, 20, and 21.

Each kind of body movements is repeated for 60 times and generates 1500 samples for each body movement. 200 samples out of 1500 samples are randomly chosen as the test data set and the rest as the training data set.

To keep the same experiment condition, the same numbers of the hidden nodes (50 hidden nodes) are applied and the same activation function is adopted (standard sigmoid function):

$$f(x) = \frac{1}{1 + e^{-x}}. \quad (21)$$

Figures 22 and 23 are the recognition accuracy and elapsed time in several different recognition algorithms for each action. As we can see, there is no significant recognition rate difference between the two algorithms, but the extreme learning machine costs less time than the other recognition algorithms in the training process.

From two aspects (see Figures 22 and 23), we could come to the conclusion that the new method in this paper has a better accuracy, efficiency, and robustness.

*4.4. The Experiment under Different Hidden Layer Node Number Conditions.* In order to verify the impact of the number of the hidden layer nodes in the extreme learning machine artificial neural network, here the experiment is carried out under different hidden layer node number conditions.

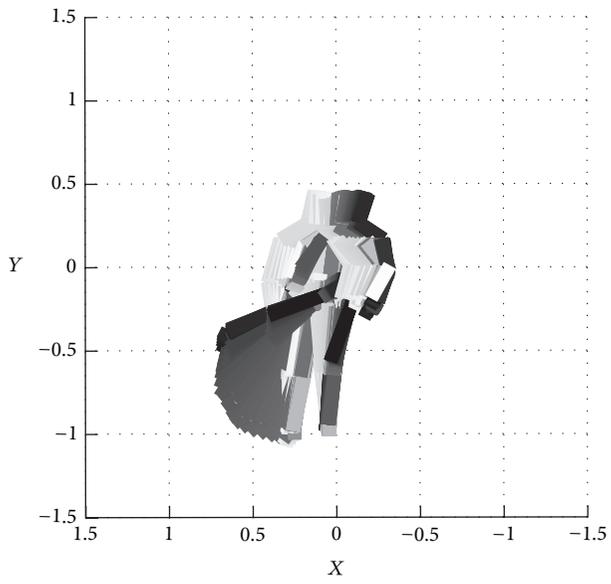
In this comparative experiment, the training time, recognition process time, and recognition accuracy in different numbers of hidden layer node condition are recorded and compared. By comparison, it is quite obvious that the extreme learning machine has a very good performance and a very fast processing speed in our application.

To keep the same experiment condition, the same experimental raw data are chosen in the second experiment and the bright normal laboratory environment. And the same activation function is selected (standard trigonometric function “Sin”).

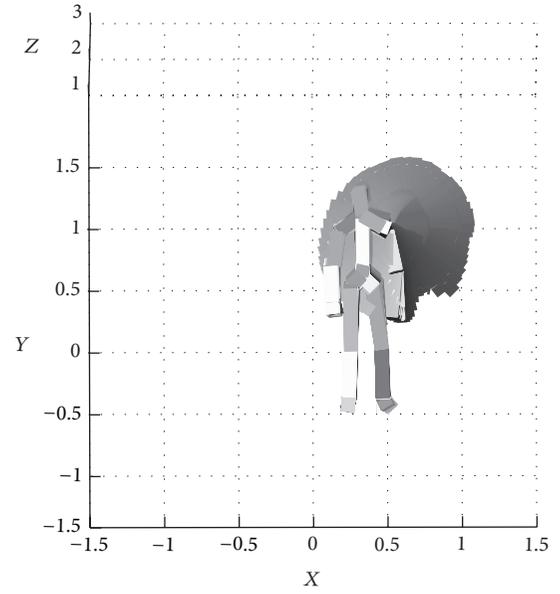
Table 1 shows that the training time is less than 1 second when the number of the hidden layer nodes is less than 300. The number of the hidden layer nodes is increased to more than 3000, the human body movement trajectory recognition process is still very fast, and the training time is less than 12 seconds, and the recognition accuracy increased. And the testing time is less than 100 milliseconds. So from this experiment, it is quite obvious that the extreme learning machine has a very good performance and a very fast processing speed.

*4.5. The Experiment under Different Training Sample Number Conditions.* In order to verify the impact of the number of the training sample on the experimental result, here some experiments are conducted under different numbers of training sample conditions.

In this comparative experiment, the training time, recognition process time, and recognition accuracy in different numbers of training sample condition are recorded and compared. From this record, it is easy to see that the extreme learning machine has a very good performance and a very



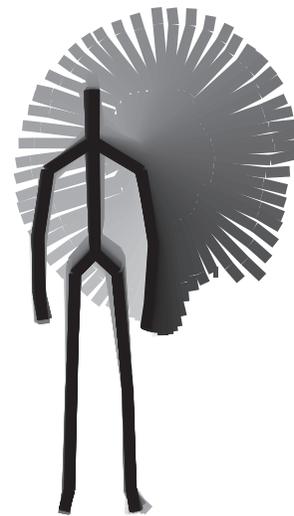
(a)



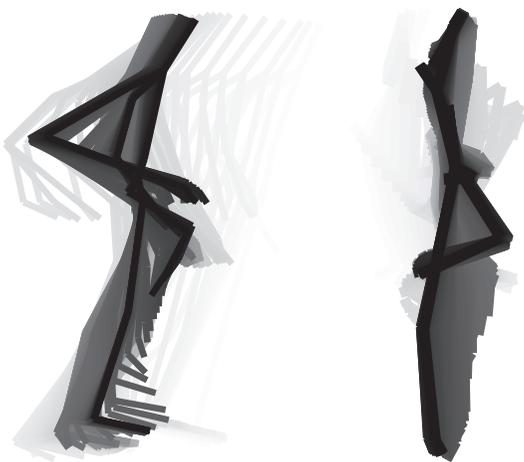
(a)



(b)



(b)



(c)

(d)

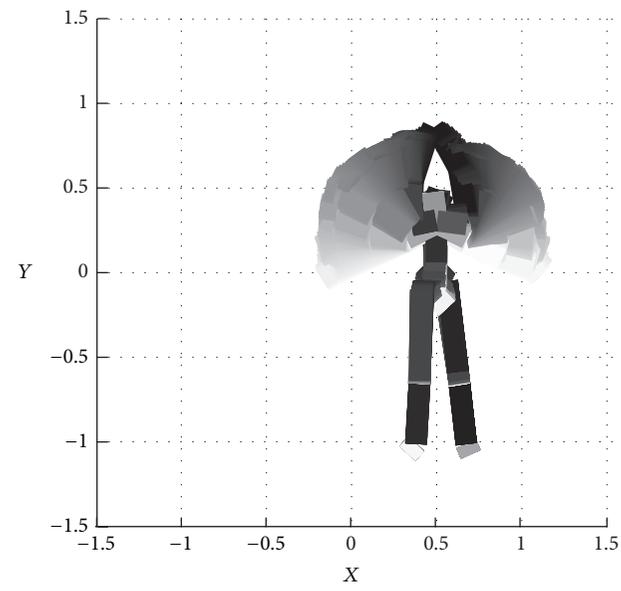


(c)

(d)

FIGURE 17: Motion history image for movement E.

FIGURE 18: Motion history image for movement F.



(a)



(b)

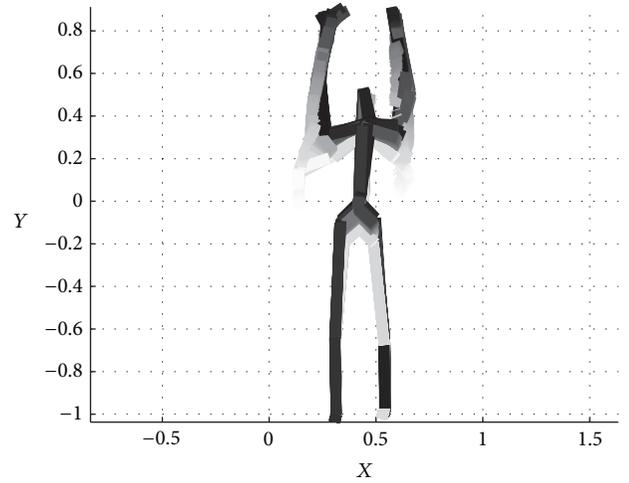


(c)



(d)

FIGURE 19: Motion history image for movement G.



(a)



(b)



(c)



(d)

FIGURE 20: Motion history image for movement H.

fast processing speed in a large quantity of training sample application.

To keep the same experiment condition, the same experimental raw data is chosen in the second experiment, the bright normal laboratory environment and 1000 hidden layer

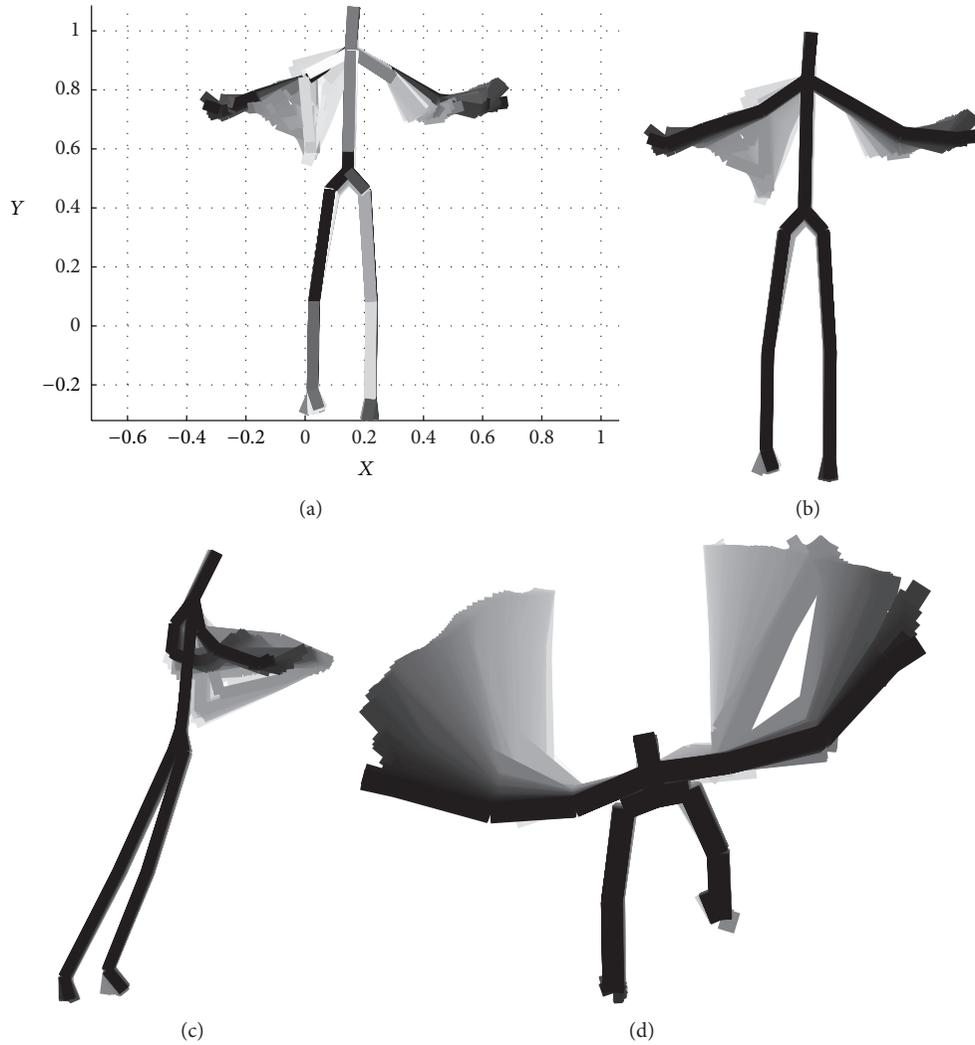


FIGURE 21: Motion history image for movement I.

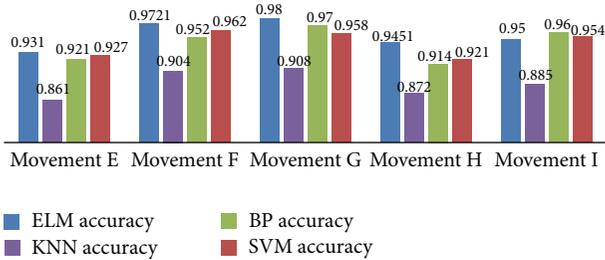


FIGURE 22: Recognition rate under different recognition algorithms.

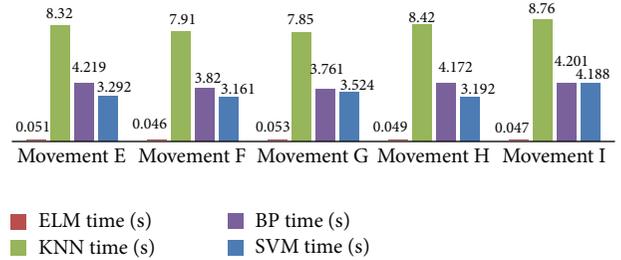


FIGURE 23: Training time under different recognition algorithms.

nodes. And the same activation function is adopted (standard trigonometric function “Sin”).

Figure 24 shows that the number of the training sample has some impact on the recognition accuracy. The human body movement trajectory accuracy improves when the number of the training sample increases. But this influence is limited; even the number of the training samples is 500; 96.5%

accuracy of the human body movement trajectory recognition is quite a good result.

4.6. *The Experiment under Different Activation Function Conditions.* In order to verify the impact of the different activation function on the experimental result, here some

TABLE 1: Different hidden layer node number experiment.

| Hidden layer node number | Training time (second) | Testing time (second) | Accuracy |
|--------------------------|------------------------|-----------------------|----------|
| 50                       | 0.0312                 | 0.00006               | 0.8550   |
| 100                      | 0.0624                 | 0.00007               | 0.8000   |
| 150                      | 0.0936                 | 0.00005               | 0.8300   |
| 200                      | 0.1248                 | 0.00003               | 0.9400   |
| 250                      | 0.2028                 | 0.00006               | 0.7850   |
| 300                      | 0.2184                 | 0.00004               | 0.9000   |
| 1000                     | 3.3540                 | 0.04690               | 0.9650   |
| 1500                     | 6.3960                 | 0.04680               | 0.9800   |
| 2000                     | 8.3939                 | 0.07800               | 0.9750   |
| 2500                     | 9.5785                 | 0.06800               | 0.9600   |
| 3000                     | 10.3741                | 0.07800               | 0.9500   |
| 3500                     | 11.2789                | 0.07800               | 0.9600   |

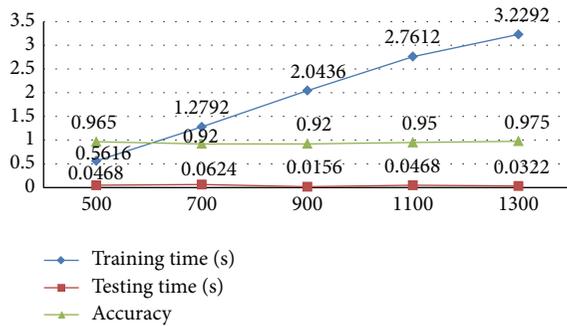


FIGURE 24: Different training sample number experiment.

experiments are carried out under different activation function conditions.

In this comparative experiment, the training time, recognition process time, and recognition accuracy in different activation function condition are recorded and compared. By comparison, it is obviously presented that activation function has a very good performance and a very fast processing speed in our application.

To keep the same experiment condition, the same experimental raw data are chosen in the second experiment, the bright normal laboratory environment and 1000 hidden layer nodes. And the 1350 training sample data and 200 testing data are adopted.

The experiments are repeated many times under the different activation functions condition, and we finally get a series of results. Figure 25 shows that different activation functions have a great impact on the recognition accuracy. The human body movement trajectory accuracy applying the standard trigonometric function “Sin” is much better than other activation functions. But the recognition efficiency using the “sigmoid” function is much better than other activation functions. So in my experiments, the “sigmoid” function is much more suitable in my application.

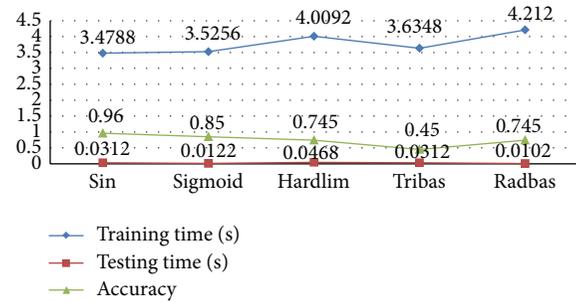


FIGURE 25: Different activation function experiment.

## 5. Conclusion

Combined with the three-dimensional motion history image and the extreme learning machine, this paper overcomes the shortcomings of the traditional body movement trajectory recognition method, such as the light, the obstruction, the background, the influence of some specific data, the heavy calculation, and the slow recognition process. The new process in this paper realizes the recognition of body movement trajectory robustly, efficiently, and accurately. Finally after some complicated experiments, it is quite obvious that the new process in the paper has better robustness, efficiency, and accuracy.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by National Nature Science Foundation of China (nos. 61272357 and 61300074) and the new century personnel plan for the Ministry of Education (NCET-10-0221).

## References

- [1] D. Weinland, R. Ronfard, and E. Boyer, “Automatic discovery of action taxonomies from multiple views,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pp. 1639–1645, New York, NY, USA, June 2006.
- [2] L. Cheng, Q. Sun, H. Su, Y. Cong, and S. Zhao, “Design and implementation of human-robot interactive demonstration system based on Kinect,” in *Proceedings of the 24th Chinese Control and Decision Conference (CCDC '12)*, pp. 971–975, Taiyuan, China, May 2012.
- [3] D. Weinland, R. Ronfard, and E. Boyer, “Free viewpoint action recognition using motion history volumes,” *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 249–257, 2006.
- [4] Y. Qi, K. Suzuki, H. Wu, and Q. Chen, “EK-means tracker: a pixel-wise tracking algorithm using kinect,” in *Proceedings of the 3rd Chinese Conference on Intelligent Visual Surveillance (IVS '11)*, pp. 77–80, Beijing, China, December 2011.

- [5] D. Weinland, R. Ronfard, and E. Boyer, "Motion history volumes for free viewpoint action recognition," in *Proceedings of the IEEE International Workshop on Modeling People and Human Interaction*, 2005.
- [6] S. B. Lee, "Real-time stereo view generation using kinect depth camera," in *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1153–1156, October 2011.
- [7] G. Huang and X. Cheng, "An automatic recognition approach of human gestures," *Journal of Southwest China Normal University*, vol. 35, no. 4, pp. 136–140, 2010.
- [8] O. Rashid, A. Al-Hamadi, and B. Michaelis, "Robust hand posture recognition with micro and macro level features using kinect," in *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems*, pp. 631–635, 2011.
- [9] S. Xu and Q. Peng, "Three-dimensional object recognition based combined moment invariants and neural network," *Computer Engineering and Applications*, vol. 44, no. 31, pp. 78–80, 2008.
- [10] J. Liu and Z. Qu, "Real-time detecting and tracking of multiple moving object based on improved motion history image," *Computer Applications*, vol. 28, no. 6, pp. 198–201, 2008.
- [11] J. Liu, M. Shah, B. Kuipers, and S. Savarese, "Cross-view action recognition via view knowledge transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pp. 3209–3216, June 2011.
- [12] Y. M. Lui, J. R. Beveridge, and M. Kirby, "Action classification on product manifolds," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 833–839, San Francisco, Calif, USA, June 2010.
- [13] F. Lv and R. Nevatia, "Single view human action recognition using key pose matching and viterbi path searching," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–8, Minneapolis, Minn, USA, June 2007.
- [14] K. Guo, P. Ishawar, and J. Konard, "Action recognition in videos by covariance matching of Sillhouette tunnels," in *Proceedings of the 22nd Brazilian Symposium on Computer Graphics and Image Processing*, pp. 299–306, 2009.
- [15] K. Guo, P. Ishwar, and J. Konrad, "Action recognition using sparse representation on covariance manifolds of optical flow," in *Proceedings of the 7th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS '10)*, pp. 188–195, IEEE, Boston, Mass, USA, August-September 2010.
- [16] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 2046–2053, San Francisco, Calif, USA, June 2010.
- [17] J. Liu and M. Shah, "Learning human actions via information maximization," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, June 2008.
- [18] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. J. Wu, "Human action recognition using non-separable oriented 3d dual-tree complex wavelet," in *Proceedings of the 9th Asian Conference on Computer Vision (ACCV '09)*, pp. 226–235, 2009.
- [19] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. M. J. Wu, "Human action recognition using extreme learning machine based on visual vocabularies," *Neurocomputing*, vol. 73, no. 10–12, pp. 1906–1917, 2010.
- [20] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [21] Y. Gu, J. Liu, Y. Chen, and X. Jiang, "Constraint online sequential extreme learning machine for lifelong indoor localization system," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '14)*, pp. 732–738, July 2014.
- [22] R. Minhas, A. A. Mohammed, and Q. M. J. Wu, "Incremental learning in human action recognition based on Snippets," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 11, pp. 1529–1541, 2012.
- [23] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.
- [24] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.
- [25] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [26] K. Schindler and L. van Gool, "Action Snippets: how many frames does human action recognition require?" in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, June 2008.
- [27] L. Yeffet and L. Wolf, "Local trinary patterns for human action recognition," in *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 492–497, Kyoto, Japan, October 2009.
- [28] X. Liu and K. Yuan, "Weight moment method based on hu invariant moments and applications," *Journal of Dalian Nationalities University*, vol. 12, no. 5, pp. 470–472, 2010.
- [29] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '04)*, pp. 985–990, Budapest, Hungary, July 2004.
- [30] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [31] J. C. A. Barata and M. S. Hussein, "The Moore-Penrose pseudoinverse: a tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, 2012.
- [32] L. Shao and R. Mattivi, "Feature detector and descriptor evaluation in human action recognition," in *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR '10)*, pp. 477–484, July 2010.

## Research Article

# An ELM-Based Approach for Estimating Train Dwell Time in Urban Rail Traffic

Wen-jun Chu, Xing-chen Zhang, Jun-hua Chen, and Bin Xu

State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China

Correspondence should be addressed to Wen-jun Chu; 12114228@bjtu.edu.cn

Received 22 July 2014; Accepted 9 October 2014

Academic Editor: Tao Chen

Copyright © 2015 Wen-jun Chu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dwell time estimation plays an important role in the operation of urban rail system. On this specific problem, a range of models based on either polynomial regression or microsimulation have been proposed. However, the generalization performance of polynomial regression models is limited and the accuracy of existing microsimulation models is unstable. In this paper, a new dwell time estimation model based on extreme learning machine (ELM) is proposed. The underlying factors that may affect urban rail dwell time are analyzed first. Then, the relationships among different factors are extracted and modeled by ELM neural networks, on basis of which an overall estimation model is proposed. At last, a set of observed data from Beijing subway is used to illustrate the proposed method and verify its overall performance.

## 1. Introduction

Dwell time is the time that a public transport vehicle spends at a station or a stop for passenger alighting and boarding [1]. In any mode of public transportation, it is an important parameter, which determines the system performance and service quality to a large extent. On one hand, dwell time constitutes a significant part of the total trip time, which is the key criterion for service quality of public transit. On the other hand, dwell time determines the capacity utilization of infrastructure, thus affecting the efficiency of the whole transit system. Therefore, reasonable estimation of dwell time plays an important role in operation of various public transit systems.

A number of studies have been conducted on dwell time estimation in various types of public transportation and corresponding research approaches can be roughly classified into two categories: regression approach and microsimulation approach.

*Regression approach* is to establish regression model with observed data to describe the relationship between dwell time and corresponding factors. This approach is first used in the estimation of bus dwell time. Levinson [2] proposed a linear regression model to estimate bus dwell time, in which the bus dwell time is formulated as a linear function

of two primary contribution factors—number of alighting and boarding passengers and the amount of time required for bus doors opening and closing. Since then, a number of studies were carried out to take into account some other contributing factors for the bus dwell time estimation. For example, Guenther and Hamat [3] investigated the relationship between the bus dwell time and bus fare collection system. Levine and Torng [4] analyzed impact of bus floor types on the bus dwell time. Jaiswal et al. [5] examined influence of platform walking on bus rapid transit stations on bus dwell time. Tirachini [6] studied impact of fare payment technology in urban bus services. Most previous studies on urban rail dwell time estimation also applied the regression approach. Weston [7] proposed a polynomial regression model using the survey data of London Metro, in which various contributing factors, including the number of alighting and boarding passengers, passenger distribution, and on-board crowdedness, are considered. Lam et al. [8] proposed a linear regression model on basis of observed data from two LRT stations. Lin and Wilson [9] compared linear and nonlinear regression models with observed data of MBTA Green Line and proved that crowdedness has a nonlinear effect on urban rail dwell time. On this basis, Puong [10] proposed a nonlinear dwell time model that can fit 90% of observed data from MBTA Red Line.

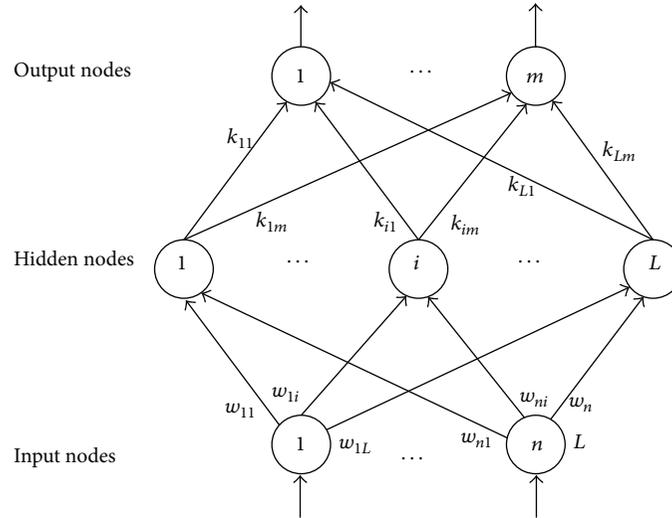


FIGURE 1: Structure of standard SLFN.

As can be seen, almost all proposed regression models on dwell time estimation are polynomial. In these studies, the model structure is first determined through certain hypothesis and then corresponding parameters are calibrated. Under this condition, though these models fit respective field data well, the generalization performance of them cannot be ensured.

*Microsimulation approach* is to calculate the required dwell time on basis of single passenger behavior description under computer environment. In recent years, computer-based pedestrian simulation technology rapidly develops and is gradually introduced into dwell time estimation. Li et al. [11] applied Monte Carlo simulation to simulate the bus dwell process, in which a binary door choice model predicting the proportion of alighting passengers through front or rear door is integrated. Zhang et al. [12] proposed a cellular automaton based alighting and boarding microsimulation model for passengers in Beijing subway stations, which is proven effective in estimating urban rail dwell time. Bae et al. [13] investigated the influence of different boarding/alighting strategies on urban rail dwell time on basis of a microsimulation model, in which an inclination function governing passengers' movement in a two-dimensional queue is introduced. In addition, some commercial pedestrian simulation software programs, such as VISSIM and Legion, are applied to calculate dwell time in many related studies.

Theoretically speaking, microsimulation models have better generalization performance than regression model. If the behavior of passengers is described properly, the model can be used in any scenario. However, existing microscopic simulation theory is still insufficient in describing pedestrian behavior under crowded condition. As a result, the accuracy of microsimulation dwell time estimation models cannot be ensured at present.

In urban rail transit system, train operation is typically based on timetables which are made in advance and the dwell time at each station is assigned beforehand. Under this condition, the reasonability of preassigned dwell time may

have a significant influence on the performance of the whole system. If the assigned dwell time is insufficient for passenger alighting and boarding, delay will happen and complicated adjustments need to be made in the predesigned timetable so as to ensure the following train operation. On the other side, if the assigned dwell time is too long, the headway between two consecutive trains will also be overlong, consequently limiting the capacity of the whole transit line. Therefore, in all urban rail transit systems, especially in those with heavy traffic such as Beijing subway, reasonable estimation of dwell time is essential to create effective timetables and make a compromise between service quality and transportation capacity.

Artificial neural network is a widely used method of data fitting. It can approximate complex nonlinear mappings directly from the input sample without making much hypothesis beforehand. In this paper, a new proposed artificial neural network method ELM is used in urban rail dwell time estimation. The outline of the paper is as follows. In Section 1, previous research regarding dwell time estimation of public transportation is reviewed. Section 2 elaborates the principles and steps of ELM. Section 3 makes a detailed analysis on the factors of train dwell time at urban rail stations and Section 4 presents the structure of the proposed model. In Section 5, several data sets on Beijing subway are used to evaluate the proposed model. Conclusions and discussions are given in Section 6.

## 2. Extreme Learning Machine

Single-hidden layer feedforward network (SLFN) is a widely used type of artificial neural network, which has been proven effective in complex nonlinear approximation [14–16]. Figure 1 illustrates the structure of a standard SLFN. In this network,  $n$  input nodes and  $m$  output nodes are included, corresponding to  $n$ -dimensional input vector and  $m$ -dimensional out vector.  $L$  nodes are contained in the hidden layer and  $\xi_i$  is the threshold of the  $i$ th hidden node.

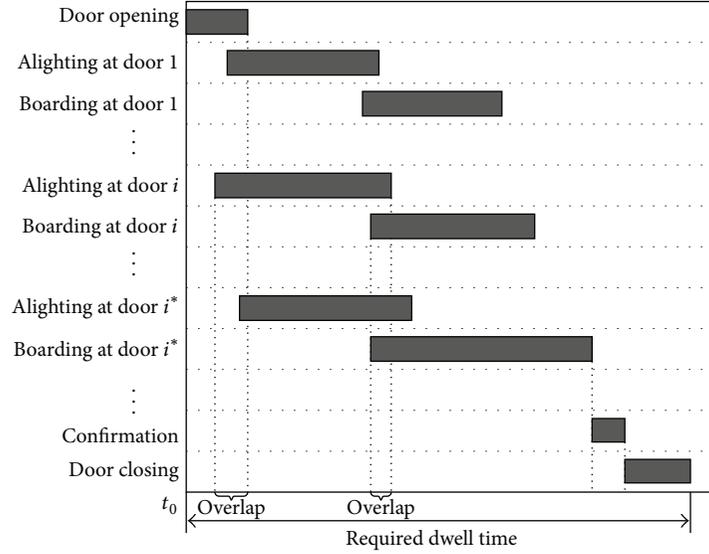


FIGURE 2: Structure of urban rail dwell time.

$g(x)$  is the activation function.  $\mathbf{w}_i = [w_{1i}, w_{2i}, \dots, w_{ni}]^T$  is the weight vector connecting the input nodes and the  $i$ th hidden node and  $\mathbf{k}_i = [k_{i1}, k_{i2}, \dots, k_{im}]^T$  is the weight vector connecting the  $i$ th hidden node and the output nodes.

Given  $N$  arbitrary training samples  $(\mathbf{x}_j, \mathbf{e}_j)$ , where  $\mathbf{x}_j = [x_{j1}, x_{j2}, \dots, x_{jn}]^T \in \mathbf{R}^n$  and  $\mathbf{e}_j = [e_{j1}, e_{j2}, \dots, e_{jm}]^T \in \mathbf{R}^m$ , the output of the above SLFN is

$$\mathbf{o}_j = \sum_i^L \mathbf{k}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \xi_i). \quad (1)$$

If this SLFN can approximate these  $N$  samples with zero error, that is,  $\sum_j^N \|\mathbf{o}_j - \mathbf{e}_j\| = 0$ , then there exist  $\mathbf{k}_i$ ,  $\mathbf{w}_i$ , and  $\xi_i$  such that

$$\sum_i^L \mathbf{k}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \xi_i) = \mathbf{e}_j. \quad (2)$$

These  $N$  equations can be written compactly as

$$\mathbf{H}\mathbf{K} = \mathbf{E}, \quad (3)$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1 \mathbf{x}_1 + \xi_1) & \dots & g(\mathbf{w}_L \mathbf{x}_1 + \xi_L) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \mathbf{x}_N + \xi_1) & \dots & g(\mathbf{w}_L \mathbf{x}_N + \xi_L) \end{bmatrix}_{N \times L}, \quad (4)$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_1^T \\ \vdots \\ \mathbf{k}_L^T \end{bmatrix}_{L \times m}, \quad (5)$$

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_N^T \end{bmatrix}_{N \times m}. \quad (6)$$

As named in Huang and Babri [17],  $\mathbf{H}$  is called the hidden layer output matrix of the SLFN and the  $i$ th column of it corresponds to the output of  $i$ th hidden node with respect to  $N$  inputs. As proven by Huang et al. [18], given arbitrary  $\mathbf{w}_i$  and  $\xi_i$ , the least square solution of  $\mathbf{K}$  in formula (3) can be obtained by formula (7):

$$\widehat{\mathbf{K}} = \mathbf{H}^\dagger \mathbf{E}, \quad (7)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ . On this basis, a simple and efficient training algorithm for SLFN called ELM is proposed [18], whose procedure can be summarized as follows.

*Step 1.* Randomly assign input weight  $\mathbf{w}_i$  and bias  $\xi_i$ ,  $i = 1, 2, \dots, L$ .

*Step 2.* Calculate the hidden layer output matrix  $\mathbf{H}$  according to formula (4).

*Step 3.* Calculate the output weight  $\mathbf{K}$  according to formula (7).

Due to the fast training speed, ELM has been widely used for many applications [19]. In this paper, ELM is applied to approximate the complex relationship between the factors of urban rail dwell time.

### 3. Factors of Urban Rail Dwell Time

Urban rail dwell time is typically defined as the time elapsed between the door opening and closing of a train sitting at a station [10]. In this period, several tasks need to be accomplished, as shown in Figure 2.

In Figure 2, the horizontal axis represents time and  $t_0$  represents the time when the train stops and doors begin to open. On the vertical axis, four types of task are listed. The duration

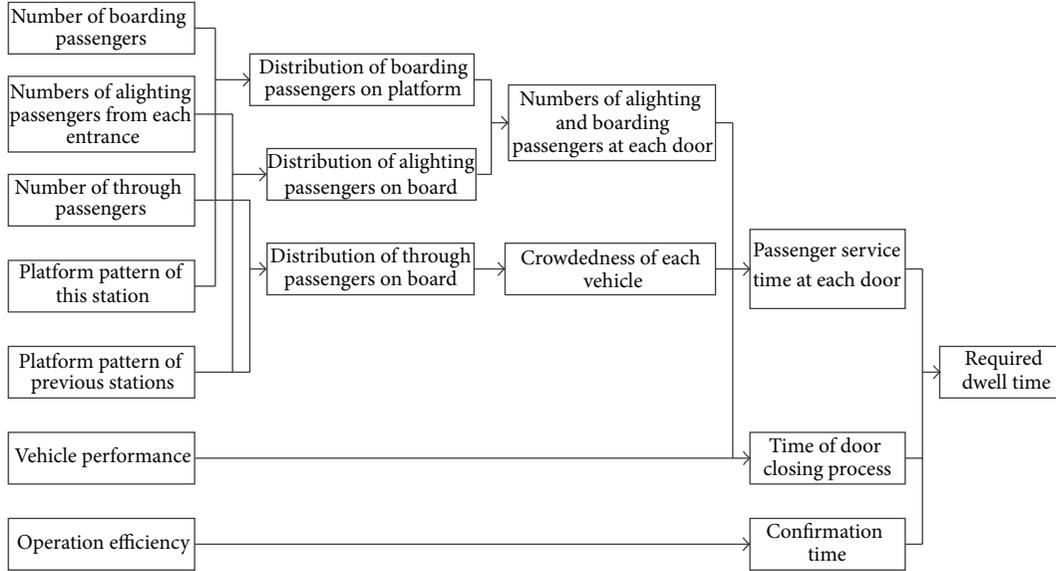


FIGURE 3: Factors of urban rail dwell time.

of door opening and closing process is mainly determined by the mechanism of the vehicles. The confirmation process represents the interval between the end of passenger alighting at all doors and the beginning of door closing process, which is used for operators confirming the completion of passenger alighting. The start time of this process depends on the door at which passenger boarding completes last, that is, the door  $i^*$ . The times of alighting and boarding tasks vary across doors. According to previous research, this is mainly because the numbers of alighting, through, and boarding passengers differ from door to door. In other words, the duration of alighting and boarding process at a door is mainly decided by the number of passengers alighting and boarding from this door and the crowdedness of corresponding vehicle. And these parameters will be affected by the passenger flow and platform pattern of this station and previous stations.

Nevertheless, in practical terms, there exist overlaps between some consecutive tasks. As shown in Figure 2, the overlap between door opening and passenger alighting represents that some passengers begin to alight before the door is fully open and the overlap between passenger alighting and boarding represents that some passengers do not obey the “get off and then on” rule. Under this condition, times of these processes cannot be separately considered, no matter from the perspective of survey or estimation. Therefore, an overall concept, passenger service time, is proposed here, which represents the period from the beginning of door opening to the end of passenger boarding at single or all doors.

On basis of the above analysis, the factors of urban rail dwell time and their interaction can be concluded, which is shown in Figure 3.

#### 4. Urban Rail Dwell Time Estimation

**4.1. Notations.** The key notations used in the dwell time estimation are shown in Notation Definitions section.

**4.2. Problem Statement.** Generally speaking, in practical operation of urban rail system, the operation-related parameters, that is, platform pattern, vehicle performance, and operation efficiency, are relatively stable. Therefore, only the influence of the traffic-related parameters which is the concern of most previous research is taken into account here. On this basis, the urban rail dwell time estimation problem can be described as follows.

Consider a  $n$ -door urban rail train that will make a stop on a station. On the train,  $A$  passengers will alight at the station and  $C$  passengers will not. On the platform of the station,  $B_j$  passengers who enter the platform through entrance  $j$  are waiting to get on this train. In addition, the train needs  $\tau_1$  to close all its doors and operators need to spend  $\tau_2$  to confirm the full close of all doors. Thus, assign a minimum dwell time  $D$  for the train, which is sufficient for passengers alighting and boarding at the station.

According to the analysis in Section 3, the required dwell time  $D$  can be seen as the accumulation of three parts: the maximum single-door passenger service time, duration of door closing process, and confirmation time; that is,

$$D = \max_i t_i + \tau_1 + \tau_2, \quad (8)$$

where the passenger service time at  $i$ th door  $t_i$  is determined by the number of boarding, alighting, and through passengers at this door; that is,

$$t_i = F(a_i, b_i, c_i). \quad (9)$$

Furthermore, for a specific station, the distribution of boarding passengers on the platform is always accorded with certain rules [18], which means certain mapping exists between the vector  $\boldsymbol{\beta} = [b_1, b_2, \dots, b_n]^T$  and the boarding passenger vector  $\mathbf{B} = [B_1, B_2, \dots, B_m]^T$ ; that is,

$$\boldsymbol{\beta} = g_1(\mathbf{B}) = g_1(B_1, B_2, \dots, B_m). \quad (10)$$

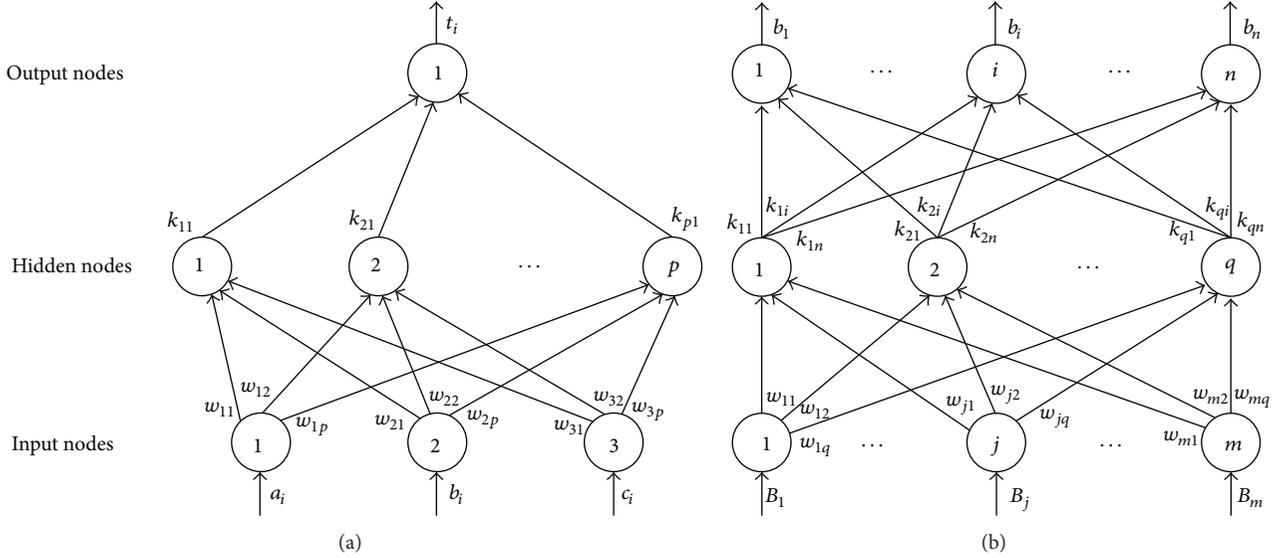


FIGURE 4: Structure of SDPST model and PPD model.

By contrast, the distribution of alighting and through passengers on board, which is determined by platform pattern of previous stations, is more complicated. In previous research, the alighting and through passengers on board are usually assumed to be uniformly distributed [10] or distributed with constant proportion [7]. In this paper, the uniform distribution is adopted for  $\alpha$  and  $\gamma$ ; that is,

$$\alpha = [a_1, a_2, \dots, a_n]^T = g_2(A) = \left[ \frac{A}{n}, \frac{A}{n}, \dots, \frac{A}{n} \right]^T, \quad (11)$$

$$\gamma = [c_1, c_2, \dots, c_n]^T = g_2(C) = \left[ \frac{C}{n}, \frac{C}{n}, \dots, \frac{C}{n} \right]^T.$$

To summarize, the required dwell time  $D$  can be described as follows:

$$D = \max_i \left\{ F \left( \frac{A}{n}, [g_1(B_1, B_2, \dots, B_m)]_{i,1}, \frac{C}{n} \right) \right\} + \tau_1 + \tau_2. \quad (12)$$

As can be seen, the key to dwell time estimation is to approximate the mappings  $F$  and  $g_1$ .

**4.3. ELM-Based Estimation Model.** In this section, two ELM neural networks are designed to approximate the mappings shown in formula (12). On this basis, an overall estimation model is proposed.

**4.3.1. Single-Door Passenger Service Time (SDPST) Model.** In order to approximate the relationship between  $t_i$  and  $(a_i, b_i, c_i)$ , that is,  $(a_i, b_i, c_i)$ , an ELM neural network is designed, whose structure is shown in Figure 4(a). As illustrated in this figure, the model has an input vector of three dimensions which represent  $a_i$ ,  $b_i$ , and  $c_i$ , respectively, and a single-dimensional output vector  $t_i$ . Sigmoid function is chosen as the activation function of the hidden nodes and the number of hidden nodes  $p$  needs to be determined through  $k$ -fold cross-validation with training data set.

**4.3.2. Platform Passenger Distribution (PPD) Model.** Another ELM neural network is designed to describe the distribution rule of passengers on platform, as shown in Figure 4(b).

This model has an input vector of  $m$  dimensions which represent the numbers of boarding passengers from each entrance and an output vector of  $n$  dimensions which represent the number of boarding passengers at each door. Besides, the activation function of this model is also sigmoid function and the number of hidden nodes is  $q$ , which also needs to be determined through cross-validation.

**4.3.3. Overall Dwell Time Estimation Model.** On basis of the previous two models, an overall model for urban rail dwell time estimation is proposed, which is shown in Figure 5. In this model, the mappings  $F$  and  $g_1$  in formula (12) are replaced by SDPST model and PPD model, respectively, and this two ELM neural networks need to be trained separately with corresponding data sets.

## 5. Model Evaluation

**5.1. Data Collection and Processing.** A survey is conducted on the outbound platform of Zhichunlu station of Line 13, Beijing subway. This platform is a typical side platform with three stairways and one escalator acting as entrances and exits, as shown in Figure 6. In the survey, 24 recorders are assigned to observe the 24 doors of trains, respectively, and another two are assigned to record the number of boarding passengers entering from the two entrances. After 10 days' survey, a raw data set containing 8304 instances from 346 trains is obtained, whose structure is illustrated in Table 1. It should be noted that the actual number of through passenger cannot be observed precisely from platform. Therefore, the attribute  $c$ , which is used to describe the crowdedness on the vehicle, is replaced by the number of through passengers that stand on board near the door.

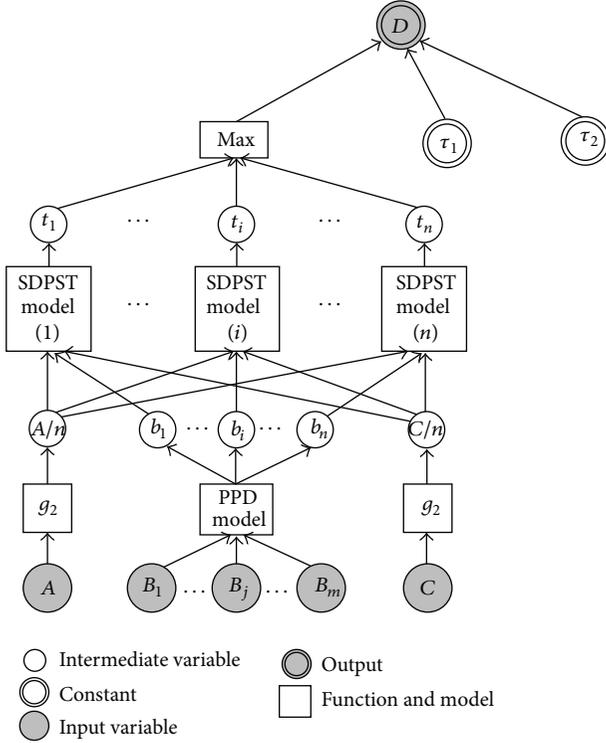


FIGURE 5: Overall dwell time estimation model.

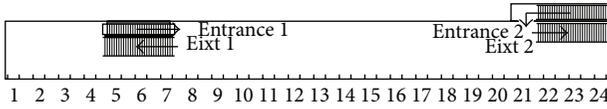


FIGURE 6: Layout of the outbound platform of Zhichunlu station (Line 13).

TABLE 1: Structure of the raw data set.

| Attribute | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| $r$       | Index of records                                                                                      |
| $i$       | Index of doors                                                                                        |
| $u$       | Index of trains                                                                                       |
| $a$       | Number of alighting passengers at single door                                                         |
| $b$       | Number of boarding passengers at single door                                                          |
| $c$       | Number of through passengers that stand on board near the door                                        |
| $B_1$     | Number of boarding passengers entering from Entrance 1                                                |
| $B_2$     | Number of boarding passengers entering from Entrance 2                                                |
| PST       | Passenger service time                                                                                |
| $T$       | Actual dwell time of train, which spans from the beginning of door opening to the end of door closing |
| $T_0$     | Scheduled dwell time of train                                                                         |

From this raw data set, the operation-related parameters and three useful data sets are derived.

**5.1.1. Operation-Related Parameters.** Firstly, the confirmation and door closing times are derived. Considering the effect of scheduled dwell time, only the records in which actual dwell time exceeds scheduled dwell time are used and the sum of constant parameters  $\tau_1$  and  $\tau_2$  is assigned with the average of differences between  $T$  and PST; that is,

$$\tau_1 + \tau_2 = \frac{\sum_{\{r|T-T_0>60\}} (T - \text{PST})}{|\{r | T - T_0 > 60\}|}. \quad (13)$$

**5.1.2. SDPST Data Set.** This data set has 8304 instances, each of which represents a passenger service process at a single door. Four attributes,  $a$ ,  $b$ ,  $c$ , and PST, are contained and corresponding data can be extracted directly from the raw data set. This data set can be used to train the SDPST model.

**5.1.3. PPD Data Set.** 346 instances are contained in this data set, each of which corresponds to an observed train. There are 26 attributes per instance. Two of them are the numbers of boarding passengers entering from the two entrances (named as  $B_1$  and  $B_2$ ) and the rest represent the number of boarding passengers at each door (named as  $b_i$ ,  $i = 1, 2, \dots, 22$ ). In this way, the distribution of boarding passengers for each observed train can be described by the instances of this data set. Therefore, this data set can be used to train the PPD model.

**5.1.4. Dwell Time Data Set.** This data set concerns the relationship between dwell time of trains and corresponding passenger flow. Therefore, 346 records corresponding to 346 observed trains are included and each of them has five attributes: the total number of alighting passengers  $A$ , the numbers of boarding passengers entering from Entrance 1 and Entrance 2, that is,  $B_1$  and  $B_2$ , the total number of through standees  $C$ , and required dwell time  $D$ . The former three attributes can all be obtained through accumulating the corresponding single-door data of the raw data set, while  $D$  is obtained according to

$$D = \begin{cases} T & T - T_0 > 60 \\ \text{PST} + \tau_1 + \tau_2 & \text{otherwise.} \end{cases} \quad (14)$$

**5.2. Training of SDPST Model.** With the SDPST data set, ELM is used to train the SDPST model. Meanwhile, for comparison, another two popular algorithms, LMBP and SVM, are also applied on this specific regression problem. All the attributes in this data set are normalized into range  $[0, 1]$  and the data set is divided into two parts: 4000 observations are used for training and the rest are used for testing. For ELM, the number of hidden nodes  $p$  is gradually increased by an interval of 5 and the optimal number 65 is obtained using 3-fold cross-validation method, which is illustrated in Figure 7. Similarly, the number of hidden nodes in the BP network is also determined through repeated cross-validations. For SVM, RBF is used as kernel function and the cost parameter and kernel parameter are both chosen from set  $\{2^{-10}, 2^{-9}, 2^{-8}, \dots, 2^9, 2^{10}\}$  through repeated tests.

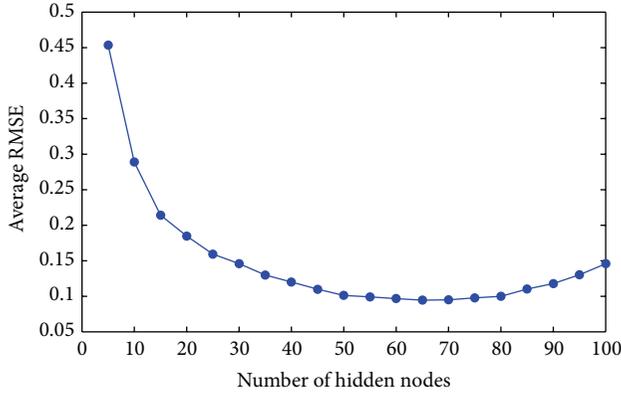


FIGURE 7: Tuning the number of hidden nodes in the ELM-based SDPST model.

TABLE 2: Comparison of performance of ELM, LMBP, and SVM on SDPST data set.

| Algorithms | Number of nodes/SVs | Time (s) |         | RMSE     |         |
|------------|---------------------|----------|---------|----------|---------|
|            |                     | Training | Testing | Training | Testing |
| ELM        | 65                  | 0.1358   | 0.4128  | 0.0865   | 0.0945  |
| LMBP       | 20                  | 2.1722   | 0.0986  | 0.0904   | 0.1218  |
| SVM        | 310.24              | 4.6375   | 1.2519  | 0.0853   | 0.1046  |

All the simulations are carried out in MATLAB 8.2 environment running in a Core2 Quad, 2.67 GHz CPU, and corresponding results are shown in Table 2. As shown in this table, no matter in training speed or generalization performance, ELM is remarkably better than the other two algorithms. In other words, the ELM-based SDPST performs better in estimating the single-door passenger service time.

For further comparison, a basic social force model [20] is established to simulate passengers alighting and boarding at single door of urban train. The parameters of this model are calibrated according to the observed data of a basic case, in which the numbers of alighting, boarding, and through passengers are all 5; that is,  $a = b = c = 5$ . On this basis, different cases are tested on this microsimulation model and the results are compared with the proposed model. In the test, the numbers of alighting and through passengers are all set to be 5; that is,  $a = c = 5$ . The number of boarding passengers is gradually increased and corresponding PST outputted by the microsimulation model is compared with the result estimated by the ELM-based SDPST model, which is shown in Figure 8. As can be seen, the results of the proposed model are in good accordance with the observed data. The microsimulation model fits the observed data well when  $b \leq 16$ , but it does not perform well when  $b > 16$ .

Furthermore, using the SDPST model trained by ELM, the relationship between passenger service time (PST) and corresponding factors ( $a$ ,  $b$ , and  $c$ ) at single door is also investigated. With the other two factors fixed at 5, the variation of PST with each factor is tested. As shown in Figure 9, PST is in nonlinear relationship with each of the

TABLE 3: Comparison of performance of ELM, BP, and SVM on PPD data set.

| Algorithms | Number of nodes/SVs | Time (s) |         | RMSE     |         |
|------------|---------------------|----------|---------|----------|---------|
|            |                     | Training | Testing | Training | Testing |
| ELM        | 25                  | 0.0057   | 0.0075  | 0.0987   | 0.1015  |
| LMBP       | 10                  | 0.1167   | 0.0029  | 0.1077   | 0.1102  |
| SVM        | 42.32               | 0.0764   | 0.0828  | 0.0972   | 0.1023  |

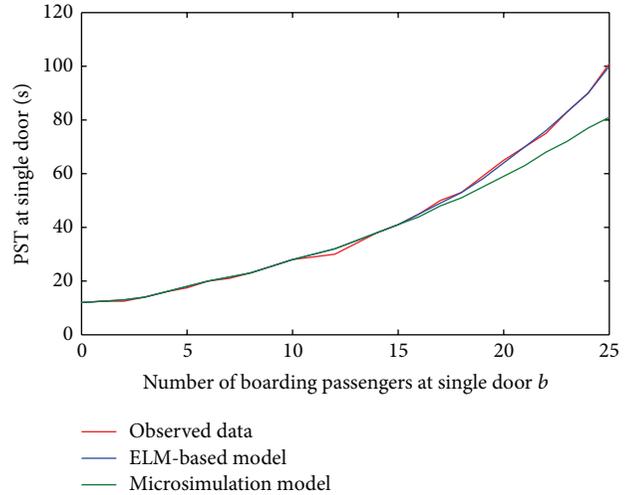


FIGURE 8: Comparison of performance of ELM-based model and microsimulation model.

three factors, which is much different with previous studies [8–10].

**5.3. Training of PPD Model.** With the PPD data set, the PPD model is trained to describe the boarding passenger distribution on the outbound platform of Zhichulu station (Line 13). The data set is also normalized into  $[0, 1]$  and divided into two parts: 200 observations are used for training and the rest are used for testing. The other two algorithms, LMBP and SVM, are also applied on this data set and their performances are compared with ELM in Table 3. As can be seen, the training speed of ELM is still remarkably faster than that of the other two algorithms. As for generalization performance, ELM is similar to the SVM and slightly better than LMBP. In conclusion, the ELM-based model obtains best performance on the PPD data set.

**5.4. Evaluation of Overall Estimation Model.** With the above two models trained by ELM, the overall model can be used to estimate the train dwell time of Line 13 at Zhichunlu station. The proposed overall model is compared with two polynomial models. One is proposed by Lam et al. [8] and

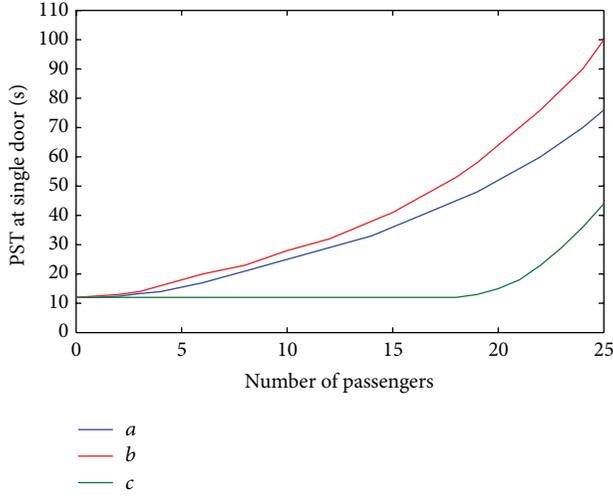


FIGURE 9: Relationship between PST and corresponding factors.

TABLE 4: Comparison of performance of proposed model and AP model.

| Models         | Coefficient of determination ( $R^2$ ) |
|----------------|----------------------------------------|
| Proposed model | 0.8972                                 |
| Lam's model    | 0.6711                                 |
| Puong's model  | 0.7802                                 |

shown as formula (15). The other is proposed by Puong [10] and shown as formula (16):

$$D = \mu_1 + \mu_2 A + \mu_3 B, \quad (15)$$

$$D = \nu_1 + \nu_2 \frac{A}{n} + \nu_3 \frac{B}{n} + \nu_4 \left(\frac{C}{n}\right)^3 \frac{B}{n}. \quad (16)$$

Using the dwell time data set, least squares method is used to calibrate the parameters of the above two models. Considering the outputs of these three models are all single-dimensional, the coefficient of determination which is usually denoted as  $R^2$  is adopted to evaluate their regression performance. The model whose  $R^2$  is closer to 1 is considered better. The results are listed in Table 4. As can be seen, the ELM-based model proposed in this paper performs much better than the other two polynomial models.

## 6. Conclusions

This paper proposed a new model to estimate urban rail dwell time. In this model, two crucial relationships among the factors of urban rail dwell time are modeled by two SLFNs, which are trained with ELM. Using a set of observed data from Beijing subway, the training of these two networks is illustrated, during which ELM is proven more effective than other two algorithms, and advantage of the proposed approach is also verified by comparing with an existing estimation model.

## Notation Definitions

- $i$ : Index of doors
- $n$ : Number of doors
- $j$ : Index of platform entrances
- $m$ : Number of platform entrances
- $A$ : Total number of alighting passengers
- $B_j$ : Number of boarding passengers entering the platform through  $j$ th entrance
- $\mathbf{B}$ :  $m$ -dimensional column vector, whose  $j$ th component is  $B_j$ ; that is,  $\mathbf{B} = [B_1, B_2, \dots, B_m]^T$
- $C$ : Total number of through passengers
- $a_i$ : Number of alighting passengers at  $i$ th door
- $b_i$ : Number of boarding passengers at  $i$ th door
- $c_i$ : Number of through passengers at  $i$ th door
- $\boldsymbol{\alpha}$ :  $n$ -dimensional column vector, whose  $i$ th component is  $a_i$ ; that is,  $\boldsymbol{\alpha} = [a_1, a_2, \dots, a_n]^T$
- $\boldsymbol{\beta}$ :  $n$ -dimensional column vector, whose  $i$ th component is  $b_i$ ; that is,  $\boldsymbol{\beta} = [b_1, b_2, \dots, b_n]^T$
- $\boldsymbol{\gamma}$ :  $n$ -dimensional column vector, whose  $i$ th component is  $c_i$ ; that is,  $\boldsymbol{\gamma} = [c_1, c_2, \dots, c_n]^T$
- $t_i$ : Passenger service time at  $i$ th door
- $\tau_1$ : Duration of door closing process
- $\tau_2$ : Confirmation time
- $D$ : Required dwell time.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors are grateful to the editor and reviewers for their valuable suggestions which improved the paper. This work is supported by National Natural Science Foundation of China (U1361114).

## References

- [1] Q. Meng and X. Qu, "Bus dwell time estimation at bus bays: a probabilistic approach," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 61–71, 2013.
- [2] H. S. Levinson, "Transit travel time performance," *Transportation Research Record*, vol. 915, pp. 1–6, 1983.
- [3] R. P. Guenther and K. Hamat, "Transit dwell time under complex fare structure," *Journal of Transportation Engineering*, vol. 114, no. 3, pp. 367–379, 1988.
- [4] J. Levine and G. Torng, "Dwell-time effects of low-floor bus design," *Journal of Transportation Engineering*, vol. 120, no. 6, pp. 914–929, 1994.
- [5] S. Jaiswal, J. Bunker, and L. Ferreira, "Influence of platform walking on brt station bus dwell time estimation: Australian analysis," *Journal of Transportation Engineering*, vol. 136, no. 12, pp. 1173–1179, 2010.

- [6] A. Tirachini, "Estimation of travel time and the benefits of upgrading the fare payment technology in urban bus services," *Transportation Research C: Emerging Technologies*, vol. 30, pp. 239–256, 2013.
- [7] J. G. Weston, "London underground train service model: a description of the model and its uses," in *Proceedings of the Computer Applications in Railway Planning and Management Conference (COMPRAIL '90)*, pp. 133–147, Rome, Italy, 1990.
- [8] W. H. K. Lam, C.-Y. Cheung, and C. F. Lam, "A study of crowding effects at the Hong Kong light rail transit stations," *Transportation Research Part A: Policy and Practice*, vol. 33, no. 5, pp. 401–415, 1999.
- [9] T. M. Lin and N. H. M. Wilson, "Dwell time relationships for light rail systems," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1361, pp. 287–295, 1991.
- [10] A. Puong, *Dwell Time Model and Analysis for the MBTA Red Line*, MIT OpenCourseWare, 2000, <http://ocw.mit.edu/index.htm>.
- [11] M. T. Li, F. Zhao, L. F. Chow, H. Zhang, and S. C. Li, "Simulation model for estimating bus dwell time by simultaneously considering numbers of disembarking and boarding passengers," *Transportation Research Record*, no. 1971, pp. 59–65, 2006.
- [12] Q. Zhang, B. Han, and D. Li, "Modeling and simulation of passenger alighting and boarding movement in Beijing metro stations," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 5, pp. 635–649, 2008.
- [13] S. Bae, F. Eshghi, S. M. Hashemi, and R. Moienfar, "Passenger boarding/alighting management in urban rail transportation," in *Proceedings of the Joint Rail Conference (JRC '12)*, pp. 823–829, Philadelphia, Pa, USA, April 2012.
- [14] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [15] G. B. Huang, *Learning capability of neural networks [Ph.D. thesis]*, Nanyang Technological University, Singapore, 1998.
- [16] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799–801, 2000.
- [17] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.
- [18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [19] R. Rajesh and J. S. Prakash, "Extreme learning machines—a review and state-of-the-art," *International Journal of Wisdom Based Computing*, vol. 1, no. 1, pp. 35–49, 2011.
- [20] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, 1995.

## Research Article

# Distance Based Multiple Kernel ELM: A Fast Multiple Kernel Learning Approach

Chengzhang Zhu,<sup>1</sup> Xinwang Liu,<sup>1</sup> Qiang Liu,<sup>1</sup> Yuewei Ming,<sup>1</sup> and Jianping Yin<sup>2</sup>

<sup>1</sup> College of Computer, National University of Defense Technology, Changsha 410073, China

<sup>2</sup> State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Chengzhang Zhu; kevin.zhu.china@gmail.com

Received 20 August 2014; Revised 7 November 2014; Accepted 10 November 2014

Academic Editor: Tao Chen

Copyright © 2015 Chengzhang Zhu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose a distance based multiple kernel extreme learning machine (DBMK-ELM), which provides a two-stage multiple kernel learning approach with high efficiency. Specifically, DBMK-ELM first projects multiple kernels into a new space, in which new instances are reconstructed based on the distance of different sample labels. Subsequently, an  $\ell_2$ -norm regularization least square, in which the normal vector corresponds to the kernel weights of a new kernel, is trained based on these new instances. After that, the new kernel is utilized to train and test extreme learning machine (ELM). Extensive experimental results demonstrate the superior performance of the proposed DBMK-ELM in terms of the accuracy and the computational cost.

## 1. Introduction

Currently, classification and regression are two major problems targeted by most of machine learning, pattern recognition, and data mining methods. For example, one may use a classifier in fingerprint identification system [1] or introduce regression to predict stock price [2], and so forth. As a kind of structure that can process classification and regression problems, single hidden layer feedforward networks (SLFNs) have been extensively studied. In previous work, many methods have been proposed to train SLFNs, such as back propagation algorithm [3] and SVM [4]. However, the above training methods may have two main drawbacks, that is, local extremum and long training time.

Recently, Huang et al. have proposed extreme learning machine (ELM) to train SLFNs in an extremely fast fashion [5, 6]. It has been proved that ELM can overcome the main drawbacks of previous SLFNs training methods. Specifically, ELM can learn a global optimal solution of the SLFNs' parameters, which can contribute to a high performance in both classification and regression problems, in an extremely short time because it only needs to train the output weights between hidden layer and output layer via the least square method [7, 8]. The reason is that ELM only needs to train

the output weights between the hidden layer and the output layer via the least square method. Another attractive feature of ELM is that it establishes a unified model for solving both classification and regression problems [8]. Considering the outstanding advantages of ELM, numerous valuable applications based on the ELM have been proposed, such as [9, 10]. Meanwhile, many researchers have promoted the evolution of ELM recently, including proposed online sequential ELM [11], voting based ELM [12], weighted ELM [13], sparse ELM [14], and kernel based ELM [8]. As it can get rid of the impact of the number of hidden nodes, one of their work, the kernel based ELM, is applicable to a wide range and has perfect performance. So far, most studies that towards kernel based ELM focus on using a single kernel. However, since description ability of single kernel is weaker than multiple kernels in most cases, it may get better results to use multiple kernels in kernel based ELM. Moreover, using multiple kernels can handle multiple source information fusing problem, which can further improve the performance of classification and regression in some cases.

Multiple kernel learning (MKL) is a kind of machine learning method that can enable classifier and regressor to utilize multiple kernel information. Given a set of base kernels, the goal of MKL is to construct a new kernel,

which can be more suitable to address the problem at hand, through learning an appropriate combination of base kernels. Typically, MKL can be sorted into two categories, one-stage approach and two-stage approach, by its approach. The one-stage approach learns the combination coefficients of the base kernels and the parameters of the classifier jointly by solving a joint optimization objective function. After [15] pioneered this kind of approach, which got great attention, a lot of work following it has been proposed, including [16–19], to name just a few. On the contrary, the two-stage approach, such as [20, 21], constructs a new kernel firstly by finding a suitable combination of base kernels and then it uses this combination in classifier or regressor.

In previous work, some researchers tried to introduce multiple kernels into ELM, such as [22, 23], and got satisfactory results. Although these efforts made pioneering achievements, all of them fail to achieve an extreme learning speed or cannot make sense in both classification and regression cases. In this paper, we propose a novel multiple kernel based ELM, named distance based multiple kernel extreme learning machine (DBMK-ELM), which is a fast two-stage multiple kernel learning approach and can be adapted to both classification and regression. In the first stage, DBMK-ELM finds the combination coefficients of pregenerated base kernels based on training samples. It first projects original base kernels into a new space and reconstructs new instances based on the distance of training samples. Then, it transfers the multiple kernel learning problem to a binary classification problem or a regression problem and solves it using the least square method. Finally, it constructs the new kernel from base kernels based on the learned combination coefficients. In the second stage, DBMK-ELM adopts the new kernel in kernel based ELM. Experimental results demonstrate the following advantages of our proposed DBMK-ELM: (1) the training time of DBMK-ELM is extremely short compared with traditional MKL methods; (2) DBMK-ELM can fully use multiple source information and outperform previous MKL methods in terms of the classification and regression accuracy; (3) DBMK-ELM can improve the robustness and the accuracy of basic kernel based ELM in both classification and regression cases.

The rest of paper is organized as follows. Section 2 briefly introduces ELM. Then, Section 3 presents the proposed DBMK-ELM. Meanwhile, Section 4 evaluates the performance of DBMK-ELM via extensive experiments. Finally, Section 5 concludes the paper.

## 2. Related Work

Our proposed method is extended from ELM, specifically, kernel based ELM. In this section, we briefly introduce ELM and kernel based ELM.

**2.1. Extreme Learning Machine.** Extreme learning machine is a perfect training method of single hidden layer feedforward networks (SLFNs). Since it was proposed by Huang et al. [6], ELM has been widely used in numerous areas. The main advantages include but are not limited to (1) the extreme training speed; (2) the fascinating generalization

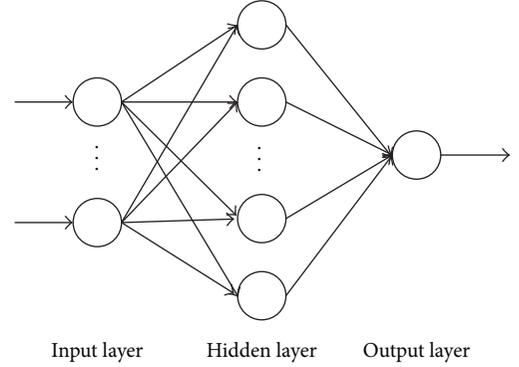


FIGURE 1: Single hidden layer feedforward networks.

performance. These advantages are attributed to the fact that ELM randomly generates the weights between input layer and hidden layer and uses a least squares method to learn the other weights. For instance, if we consider a SLFNs as Figure 1 illustrating, which has  $L$  hidden layer nodes and one output layer node, the output function of it is as follows:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \boldsymbol{\beta}, \quad (1)$$

where  $\boldsymbol{\beta}$  is the weights between hidden layer and output layers and  $\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]$  is the value vector of hidden layer that maps original features into a new feature space. Different from other SLFNs training methods, ELM can use any  $\mathbf{h}(\mathbf{x})$ , in which  $G(\mathbf{a}_i, b_i, \mathbf{x})$  is a nonlinear piecewise continuous function, such as sigmoid function and Gaussian function. Moreover,  $\mathbf{a}$ , the weights between input layer and hidden layer, and  $b$ , the bias of hidden nodes, can be randomly chosen based on any continuous probability distribution in ELM. Thus, the only parameter that ELM must learn is  $\boldsymbol{\beta}$ . ELM learns  $\boldsymbol{\beta}$  through solving the following optimization problem [8]:

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{\beta}\|^2 + \frac{1}{2} C \sum_{i=1}^N \xi_i^2 \quad (2)$$

$$\text{s.t. } \mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta} = y_i - \xi_i, \quad i = 1, \dots, N,$$

where  $\{\mathbf{x}_i, y_i\}$  is training samples. According to KKT theorem,  $\boldsymbol{\beta}$  can be calculated from

$$\boldsymbol{\beta} = \left( \frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{Y} \quad (3)$$

or

$$\boldsymbol{\beta} = \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{Y}, \quad (4)$$

where  $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_N)]^T$  and  $\mathbf{Y} = [y_1, \dots, y_N]^T$ . One can use (3) for the case where the number of training samples is huge, that is,  $N \gg L$ , and use (4) on the contrary, that is,  $N \ll L$ .

2.2. *Kernel Based Extreme Learning Machine.* Kernel method can be easily introduced into ELM. Specifically, ELM kernel  $K$  can be derived from ELM feature mapping function  $\mathbf{h}(\mathbf{x})$  as follows:

$$\mathbf{K} = \mathbf{H}\mathbf{H}^\top : K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j). \quad (5)$$

Therefore, ELM output function can be written as follows:

$$f(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^\top \left( \frac{\mathbf{I}}{C} + \mathbf{K} \right)^{-1} \mathbf{Y}. \quad (6)$$

In this case, users do not have to know the  $\mathbf{h}(\mathbf{x})$  or set the number of hidden nodes  $L$ , that is, the dimension of ELM feature space.

### 3. Distance Based Multiple Kernel ELM

Since the goal of multiple kernel learning is to construct a new kernel that more suitable for problem processing, the nature of “good” kernel must be considered. Generally, kernel can be seen as a measure of similarity. Each entry in a kernel matrix represents a similarity of two corresponding samples. From this point of view, a “good” kernel can display the true similarity of sample pairs. In other words, if two sample pairs have similar similarity, their corresponding value in the “good” kernel will also be similar. To this end, we propose distance based multiple kernel ELM. It measures similarities of sample pairs by their “label distance” that will be defined in the following part and uses this information to construct a “good” kernel. DBMK-ELM is a kind of two-stage multiple kernel learning method. In the first stage, it learns a new kernel. In the second stage, it uses the new kernel in kernel based ELM.

3.1. *Label Distance.* We first define the “label distance”. As a significant part of training samples, the label contains class information in the classification case and dependent variable information in the regression case, which can be used to measure true similarity between samples. Considering different label meaning between classification and regression, we discuss “label distance” in these two cases separately.

In the classification case, the label means the class that a sample belongs to and samples can be seen as similar when they are in the same class. In other words, if two samples have the same label, they can be seen as similar. On the contrary, if two samples have different labels, they can be seen as different. However, in this case, it is difficult to discriminate how different two samples are, because the difference between classes is not clear. Therefore, label distance is defined to 0 if two samples have the same label and defined to 1 if two samples have different labels. Formally, we define label distance  $g(y, y')$  as follows:

$$g(y, y') = \begin{cases} 0 & y = y' \\ 1 & y \neq y' \end{cases}. \quad (7)$$

In the regression case, the label means the value of the dependent variable. Typically, the similarity of samples

is directly represented in the difference of their values of dependent variable in regression cases, for example, pollution prediction, housing number prediction, and stock price prediction, to name just a few. Thus, label distance can be defined as distance between two values of the dependent variable. Admittedly, various measurements can be used to measure this distance, but Euclidean distance is used in this paper. We, in the regression case, formally define the label distance  $g(y, y')$  as follows:

$$g(y, y') = |y - y'|. \quad (8)$$

3.2. *Multiple Kernel Learning Based on Distance.* Since label distance has been defined above, the distance information can be used to guide the new kernel learning. In this subsection, we show how does DBMK-ELM perform multiple kernel learning based on distance. As we discussed, the new optimal kernel can be seen as a linear combination of base kernels. Therefore, the goal of distance based multiple kernel learning (DBMK) is to learn the combination coefficients of base kernels from which a new kernel that each entry value is coincident to the label distance of the corresponding sample pair can be generated. Considering values of the same entry in each base kernels, corresponding to the same sample pair, if they are seen as input features and the label distance of the same sample pair is seen as output value, the DBMK can be transformed to the regression problem, in which parameters of the regressor are the combination coefficients that need to be learned. To this end, DBMK learns the combination coefficients following the next two steps. Firstly, it reconstructs a new sample space, named K-space, in which each sample corresponds to a sample pair in the original sample space, based on multiple base kernels and label distance of sample pairs. Secondly, it solves a regression problem in this new space to find combination coefficients of kernels. After this two steps, DBMK-ELM can obtain the new optimal kernel through combining the base kernels using learned combination coefficients.

For machine learning problems, including classification and regression, if training samples  $(\mathbf{x}, y)$  are drawn from a distribution  $P$  over  $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times \mathbb{R}$  and  $p$  base kernels, which must satisfy the positive semi-definite condition, are generated by a set of kernel functions  $\{k_1(\cdot, \cdot), \dots, k_p(\cdot, \cdot)\}$  and denoted as  $\mathbf{K}_1, \dots, \mathbf{K}_p$  with  $\mathbf{K}_i: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ , DMKL-ELM reconstructs the K-space as  $\{\mathbf{z}_{\mathbf{x}\mathbf{x}'}, t_{yy'} \mid ((\mathbf{x}, y), (\mathbf{x}', y')) \sim P \times P \subset \mathbb{R}^P \times \mathbb{R}\}$  where

$$\begin{aligned} \mathbf{z}_{\mathbf{x}\mathbf{x}'} &= (k_1(\mathbf{x}, \mathbf{x}'), \dots, k_p(\mathbf{x}, \mathbf{x}')), \\ t_{yy'} &= g(y, y'). \end{aligned} \quad (9)$$

In the K-space, DBMK-ELM learns the combination coefficients through solving a regression problem. In this problem, the training samples are  $(\mathbf{z}, t)$ , the whole samples in K-space, in which input feature vector is  $\mathbf{z}$  and the output label is  $t$ . Though numbers of methods can be used to solve the regression problem, DBMK-ELM applies an  $\ell_2$ -norm regularization least squares linear regression, which is similar to ELM output weight learning, to solve it, considering a fast



**Input:**The set of training samples,  $(\mathbf{X}_{\text{trn}}, \mathbf{Y}_{\text{trn}})$ ;The set of testing samples,  $(\mathbf{X}_{\text{tst}}, \mathbf{Y}_{\text{tst}})$ ;**Output:**the output vector of DBMK-ELM,  $\mathbf{o}$ ;

- (1) Pre-generate several different positive semi-definite base kernels  $K_1, \dots, K_p$  for training samples  $(\mathbf{X}_{\text{trn}}, \mathbf{Y}_{\text{trn}})$ ;
- (2) Reconstruct new sample space for  $\mathbf{K}_1, \dots, \mathbf{K}_p$  using (9) and get new samples  $(\mathbf{Z}, t)$ ;
- (3) Learn base kernel combination coefficients  $\boldsymbol{\mu}$  by (13) or (14);
- (4) Construct the new optimal kernel by (15) and (16);
- (5) Calculate the output  $\mathbf{o}$  using (17) for each sample  $(\mathbf{x}_i, y_i)$  in testing set  $(\mathbf{X}_{\text{tst}}, \mathbf{Y}_{\text{tst}})$ ;
- (6) **return**  $\mathbf{o}$ ;

ALGORITHM 1: Our proposed DBMK-ELM.

TABLE 1: Summary of the classification problems data sets.

| Datasets   | Number of training set | Number of testing set | Number of features | Number of classes |
|------------|------------------------|-----------------------|--------------------|-------------------|
| Ionosphere | 234                    | 117                   | 34                 | 2                 |
| Sonar      | 138                    | 70                    | 60                 | 2                 |
| wdbc       | 379                    | 190                   | 30                 | 2                 |
| wpbc       | 129                    | 65                    | 33                 | 2                 |
| Breast     | 70                     | 36                    | 9                  | 6                 |
| Glass      | 142                    | 72                    | 9                  | 6                 |
| Wine       | 118                    | 60                    | 13                 | 3                 |

*breast*, *glass*, and *wine* from UCI Machine Learning Repository [24]. Table 1 shows the number of training samples, testing samples, features, and classes in these data sets.

The regression benchmark data sets used in this experiment are taken from UCI Machine Learning Repository [24] and Statlib [25]. These sets include *PM10* [25], *bodyfat* [25], *housing* [24], *pollution* [25], *spacega* [25], *servo* [24], and *yacht* [24]. We display the information, including the number of training samples, testing samples, and features of these sets in Table 2.

Three multiple kernel classification benchmarks from bioinformatics data sets are selected in our experiment. The first of them is the original *plant* data set of TargetP [26]. The others are *PsortPos* and *PsortNeg* [27] that both for bacterial protein locations problem. We show the number of training samples, testing samples, kernels, and classes in these data sets on Table 3.

For each data set, we randomly select two-thirds of the data samples as training data and the rest as testing data. We repeat this procedure 20 times for each data set and obtain 20 partitions of original data. All algorithms in the experiment are evaluated on each partition and the averaged results are reported for each benchmark.

**4.2. Parameters Setting and Evaluation Criteria.** For both classification and regression benchmark data sets, we generate 23 kernels on full feature vector, including 20 Gaussian kernels ( $e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$ ) with  $\gamma = \{2^{-10}, 2^{-9}, \dots, 2^9\}$ , 3 polynomial

TABLE 2: Summary of the regression problems data sets.

| Datasets  | Number of training set | Number of testing set | Number of features |
|-----------|------------------------|-----------------------|--------------------|
| PM10      | 333                    | 167                   | 7                  |
| Bodyfat   | 168                    | 84                    | 14                 |
| Housing   | 337                    | 169                   | 13                 |
| Pollution | 40                     | 50                    | 15                 |
| Spacega   | 666                    | 333                   | 6                  |
| Servo     | 111                    | 56                    | 4                  |
| Yacht     | 205                    | 103                   | 6                  |

TABLE 3: Summary of the multiple kernel classification benchmark data sets.

| Datasets | Number of training set | Number of testing set | Number of kernels | Number of classes |
|----------|------------------------|-----------------------|-------------------|-------------------|
| Plant    | 627                    | 313                   | 69                | 4                 |
| PsortPos | 361                    | 180                   | 69                | 4                 |
| PsortNeg | 963                    | 481                   | 69                | 5                 |

kernels of degrees 1, 2, and 3. For the kernel based ELM [8], we test all the 23 kernels generated above and display the best result of them in our experiments according to the testing accuracy. For all algorithms, the regulation parameter  $C$  is selected from  $\{10^{-1}, 10^0, \dots, 10^3\}$  via 3-fold cross validation on training data.

We select accuracy and computational efficiency as the performance evaluation criteria. The accuracy means the classification accuracy rate in testing data for classification problems or the mean square error (MSE) in testing data for regression problems. In addition, for the regression problem, sample labels have been normalized to  $[-1, 1]$ . For all cases, the computational efficiency is evaluated by the training time.

The reported results for each benchmark include the mean value and the standard deviation of criteria in 20 partitions. In order to measure the statistical significance for the accuracy improvement, we further use the *paired student's t-test*, in which  $P$  value means the probability that

TABLE 4: Classification case: classification accuracy (%). Boldface means no statistical difference from the best one ( $P$  val  $\geq 0.05$ ).

| Data       | DBMK-ELM                                     | SimpleMKL [16]                               | $\ell_1$ -MK-ELM [23]                        | R-MK-ELM [23]                                | ELM [8]                                      | UW                                                |
|------------|----------------------------------------------|----------------------------------------------|----------------------------------------------|----------------------------------------------|----------------------------------------------|---------------------------------------------------|
| Ionosphere | <b>94.23 <math>\pm</math> 1.44</b><br>(0.25) | <b>94.23 <math>\pm</math> 1.96</b><br>(0.22) | <b>94.62 <math>\pm</math> 1.78</b><br>(1.00) | <b>94.32 <math>\pm</math> 1.82</b><br>(0.23) | 90.38 $\pm$ 2.88<br>(0.00)                   | 93.33 $\pm$ 1.93<br>(0.00)                        |
| Sonar      | <b>84.71 <math>\pm</math> 5.35</b><br>(1.00) | <b>84.36 <math>\pm</math> 5.51</b><br>(0.62) | <b>84.64 <math>\pm</math> 5.75</b><br>(0.90) | 82.57 $\pm$ 5.22<br>(0.00)                   | <b>84.21 <math>\pm</math> 4.71</b><br>(0.56) | 81.57 $\pm$ 5.16<br>(0.00)                        |
| wdbc       | <b>97.13 <math>\pm</math> 1.32</b><br>(0.70) | <b>97.05 <math>\pm</math> 1.11</b><br>(0.44) | <b>97.16 <math>\pm</math> 1.19</b><br>(0.74) | <b>97.08 <math>\pm</math> 1.45</b><br>(0.49) | <b>97.21 <math>\pm</math> 1.26</b><br>(1.00) | <b>97.11 <math>\pm</math> 1.41</b><br>(0.62)      |
| wdbc       | <b>77.38 <math>\pm</math> 3.57</b><br>(1.00) | 75.31 $\pm$ 3.55<br>(0.01)                   | 75.23 $\pm$ 3.56<br>(0.00)                   | 75.46 $\pm$ 3.91<br>(0.02)                   | <b>76.23 <math>\pm</math> 3.72</b><br>(0.16) | <b>76.31 <math>\pm</math> 4.16</b><br>(0.18)      |
| Breast     | <b>69.03 <math>\pm</math> 5.65</b><br>(1.00) | 65.56 $\pm$ 7.45<br>(0.01)                   | 65.83 $\pm$ 8.26<br>(0.02)                   | 67.08 $\pm$ 7.34<br>(0.04)                   | 65.83 $\pm$ 7.55<br>(0.03)                   | 66.94 $\pm$ 6.98<br>(0.00)                        |
| Glass      | <b>71.04 <math>\pm</math> 4.31</b><br>(1.00) | 54.86 $\pm$ 4.22<br>(0.00)                   | 68.61 $\pm$ 5.71<br>(0.01)                   | <b>70.14 <math>\pm</math> 4.33</b><br>(0.15) | 67.57 $\pm$ 5.18<br>(0.01)                   | <b>69.31 <math>\pm</math> 5.46</b><br>(0.08)      |
| Wine       | <b>97.92 <math>\pm</math> 1.94</b><br>(0.20) | <b>98.33 <math>\pm</math> 1.71</b><br>(1.00) | <b>98.17 <math>\pm</math> 2.22</b><br>(0.61) | <b>98.00 <math>\pm</math> 2.45</b><br>(0.41) | 97.58 $\pm$ 2.13<br>(0.02)                   | <b>98.00 <math>\pm</math> 2.27</b><br><b>0.43</b> |
| AVG        | <b>84.49</b>                                 | 81.39                                        | 83.47                                        | 83.52                                        | 82.72                                        | 83.22                                             |

TABLE 5: Classification case: classification training time (s).

| Data       | DBMK-ELM            | SimpleMKL [16]        | $\ell_1$ -MK-ELM [23] | R-MK-ELM [23]       | ELM [8]             | UW                  |
|------------|---------------------|-----------------------|-----------------------|---------------------|---------------------|---------------------|
| Ionosphere | 0.0210 $\pm$ 0.0016 | 0.8216 $\pm$ 0.4888   | 0.4115 $\pm$ 0.0558   | 1.9217 $\pm$ 0.0419 | 0.0009 $\pm$ 0.0001 | 0.0009 $\pm$ 0.0001 |
| Sonar      | 0.0063 $\pm$ 0.0006 | 0.2919 $\pm$ 0.1673   | 0.2498 $\pm$ 0.0396   | 0.7698 $\pm$ 0.0242 | 0.0003 $\pm$ 0.0001 | 0.0003 $\pm$ 0.0000 |
| wdbc       | 0.0580 $\pm$ 0.0051 | 4.2746 $\pm$ 4.2881   | 0.9743 $\pm$ 0.2876   | 6.4733 $\pm$ 0.2250 | 0.0023 $\pm$ 0.0004 | 0.0024 $\pm$ 0.0004 |
| wdbc       | 0.0061 $\pm$ 0.0006 | 0.3777 $\pm$ 0.3230   | 0.0685 $\pm$ 0.0303   | 0.6542 $\pm$ 0.0202 | 0.0004 $\pm$ 0.0001 | 0.0004 $\pm$ 0.0001 |
| Breast     | 0.0016 $\pm$ 0.0004 | 13.5007 $\pm$ 16.8849 | 0.1701 $\pm$ 0.0405   | 0.3024 $\pm$ 0.0214 | 0.0003 $\pm$ 0.0002 | 0.0002 $\pm$ 0.0001 |
| Glass      | 0.0083 $\pm$ 0.0016 | 6.5990 $\pm$ 16.2960  | 0.2948 $\pm$ 0.1141   | 0.7010 $\pm$ 0.0631 | 0.0005 $\pm$ 0.0001 | 0.0004 $\pm$ 0.0001 |
| Wine       | 0.0063 $\pm$ 0.0014 | 1.2676 $\pm$ 0.7903   | 0.2299 $\pm$ 0.0387   | 0.6000 $\pm$ 0.0168 | 0.0004 $\pm$ 0.0002 | 0.0004 $\pm$ 0.0001 |
| AVG        | 0.0153              | 3.8762                | 0.3427                | 1.632               | 0.0007              | 0.0007              |

two compared sets come from distributions with an equal mean. Typically, if the  $P$  value less than 0.05, the compared sets are considered having statistically significant difference.

**4.3. Classification Performance.** The classification accuracy of different methods is shown in Table 4. The content in Table 4 has following meanings, the first part is the mean  $\pm$  standard deviation and the second part is the  $P$  value calculated by the paired Student's  $t$ -test. The bold value in each cell of Table 4 represents the highest accuracy and those having no significant difference compared with the highest one. We also show the classification training time in Table 5, which presents as the mean  $\pm$  standard deviation.

As we can see from Table 4, DBMK-ELM achieves the highest correct classification rate or has no significant different compared with the best one. Meanwhile, the results in Table 5 prove that the time cost of this approach is significantly lower than SimpleMKL,  $\ell_1$ -MK-ELM, and R-MK-ELM.

**4.4. Regression Performance.** The regression accuracy of different methods is shown in Table 6 with the same representation of Table 4. And the regression training time is shown in Table 7.

In this case, we can see DBMK-ELM has the significant highest regression accuracy compared with other methods. From the time cost point of view, this situation is similar to the classification problem; that is, DBMK-ELM dramatically improved training time compared to SimpleMKL.

**4.5. Multiple Kernel Classification Benchmark Performance.** The classification accuracy for multiple kernel classification benchmarks of DBMK-ELM and other methods is shown in Table 8. And the multiple kernel classification training time is shown in Table 9. From the results, we can see DBMK-ELM significantly better than ELM. That means DBMK-ELM has the ability to perform multisource data fusion, thereby improving the performance of ELM. The DBMK-ELM is better than the state-of-the-art multiple kernel extreme learning machine in this case regarding the classification accuracy and the training time.

**4.6. Parameter Sensitivity Test.** In our proposed DBMK-ELM, there are two regularization parameters need to be set. In order to describe more clearly, we use  $C1$  and  $C2$  represents the regularization parameter in ELM training and multiple kernel learning, respectively. We choose classification data set *ionosphere* and regression data set *yacht* to test parameter sensitivity. For each data set, we set a wide range of  $C1$  and

TABLE 6: Regression case: regression accuracy (%). Boldface means no statistical difference from the best one ( $P$  val  $\geq 0.05$ ).

| Data      | DBMK-ELM                                     | SimpleMKL [16]             | ELM [8]                                      | UW                                           |
|-----------|----------------------------------------------|----------------------------|----------------------------------------------|----------------------------------------------|
| PM10      | <b>0.3242 <math>\pm</math> 0.0171 (1.00)</b> | 0.3347 $\pm$ 0.0152 (0.00) | 0.3305 $\pm$ 0.0172 (0.01)                   | <b>0.3256 <math>\pm</math> 0.0167 (0.46)</b> |
| Bodyfat   | <b>0.0559 <math>\pm</math> 0.0227 (1.00)</b> | 0.0898 $\pm$ 0.0264 (0.00) | 0.0755 $\pm$ 0.0254 (0.00)                   | 0.1321 $\pm$ 0.0246 (0.00)                   |
| Housing   | <b>0.1618 <math>\pm</math> 0.0256 (1.00)</b> | 0.2006 $\pm$ 0.0430 (0.00) | 0.1760 $\pm$ 0.0264 (0.00)                   | 0.2149 $\pm$ 0.0323 (0.00)                   |
| Pollution | <b>0.2738 <math>\pm</math> 0.0469 (1.00)</b> | 0.3167 $\pm$ 0.0660 (0.00) | <b>0.2767 <math>\pm</math> 0.0438 (0.61)</b> | 0.3110 $\pm$ 0.0606 (0.00)                   |
| Servo     | <b>0.1926 <math>\pm</math> 0.0408 (1.00)</b> | 0.2121 $\pm$ 0.0528 (0.04) | 0.2046 $\pm$ 0.0396 (0.01)                   | 0.2567 $\pm$ 0.0379 (0.00)                   |
| Spacega   | <b>0.1415 <math>\pm</math> 0.0073 (1.00)</b> | 0.1735 $\pm$ 0.0136 (0.00) | 0.1655 $\pm$ 0.0104 (0.00)                   | 0.1633 $\pm$ 0.0106 (0.00)                   |
| Yacht     | <b>0.1083 <math>\pm</math> 0.0192 (1.00)</b> | 0.1947 $\pm$ 0.0323 (0.00) | 0.1839 $\pm$ 0.0228 (0.00)                   | 0.2420 $\pm$ 0.0342 (0.00)                   |
| AVG       | <b>0.1797</b>                                | 0.2174                     | 0.2018                                       | 0.2351                                       |

TABLE 7: Regression case: regression training time (s).

| Data      | DBMK-ELM            | SimpleMKL [16]        | ELM [8]             | UW                  |
|-----------|---------------------|-----------------------|---------------------|---------------------|
| PM10      | 0.0430 $\pm$ 0.0021 | 7.3150 $\pm$ 4.2744   | 0.0010 $\pm$ 0.0002 | 0.0010 $\pm$ 0.0002 |
| Bodyfat   | 0.0088 $\pm$ 0.0009 | 5.4048 $\pm$ 3.9174   | 0.0004 $\pm$ 0.0001 | 0.0004 $\pm$ 0.0001 |
| Housing   | 0.0451 $\pm$ 0.0028 | 18.3088 $\pm$ 11.7929 | 0.0011 $\pm$ 0.0002 | 0.0010 $\pm$ 0.0002 |
| Pollution | 0.0004 $\pm$ 0.0000 | 0.0829 $\pm$ 0.0384   | 0.0001 $\pm$ 0.0000 | 0.0001 $\pm$ 0.0000 |
| Servo     | 0.0036 $\pm$ 0.0004 | 1.6819 $\pm$ 1.0180   | 0.0003 $\pm$ 0.0003 | 0.0002 $\pm$ 0.0000 |
| Spacega   | 0.1954 $\pm$ 0.0040 | 42.0023 $\pm$ 54.0406 | 0.0058 $\pm$ 0.0008 | 0.0058 $\pm$ 0.0008 |
| Yacht     | 0.0152 $\pm$ 0.0012 | 4.8831 $\pm$ 2.3245   | 0.0005 $\pm$ 0.0001 | 0.0005 $\pm$ 0.0001 |
| AVG       | 0.0445              | 11.3827               | 0.0013              | 0.0013              |

TABLE 8: Multiple kernel classification case: classification accuracy (%). Boldface means no statistical difference from the best one ( $P$  val  $\geq 0.05$ ).

| Data     | DBMK-ELM                                  | SimpleMKL [16]          | $\ell_1$ -MK-ELM [23]   | R-MK-ELM [23]           | ELM [8]                 | UW                      |
|----------|-------------------------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| Plant    | <b>91.82 <math>\pm</math> 1.43 (1.00)</b> | 67.38 $\pm$ 3.43 (0.00) | 58.85 $\pm$ 2.96 (0.00) | 85.32 $\pm$ 2.56 (0.00) | 78.51 $\pm$ 2.23 (0.00) | 74.79 $\pm$ 2.55 (0.00) |
| PsortPos | <b>87.92 <math>\pm</math> 2.03 (1.00)</b> | 80.22 $\pm$ 2.91 (0.00) | 70.31 $\pm$ 3.35 (0.00) | 84.14 $\pm$ 2.12 (0.00) | 80.83 $\pm$ 2.45 (0.00) | 81.03 $\pm$ 2.69 (0.00) |
| PsortNeg | <b>91.52 <math>\pm</math> 0.86 (1.00)</b> | 84.80 $\pm$ 1.74 (0.00) | 73.78 $\pm$ 1.82 (0.00) | 89.75 $\pm$ 1.23 (0.00) | 85.87 $\pm$ 1.81 (0.00) | 87.31 $\pm$ 1.42 (0.00) |
| AVG      | <b>90.42</b>                              | 77.47                   | 67.64                   | 86.40                   | 81.74                   | 81.04                   |

C2. Specifically, we have used 10 different values of  $C1$  and 10 different values of  $C2$  from  $\{10^{-1}, 10^0, \dots, 10^7, 10^8\}$ . For each  $(C1, C2)$  pair, we repeat 20 times on each data set to get the average accuracy. The result of classification case and regression case is shown in Figures 2 and 3, respectively. As can be seen from the results, the performance of DBMK-ELM is not sensitivity while  $C1$  and  $C2$  vary within a wide range.

**4.7. Discussion.** The experimental results have illustrated that DBMK-ELM can achieve a high accuracy with a fast learning speed. However, two issues need to be discussed.

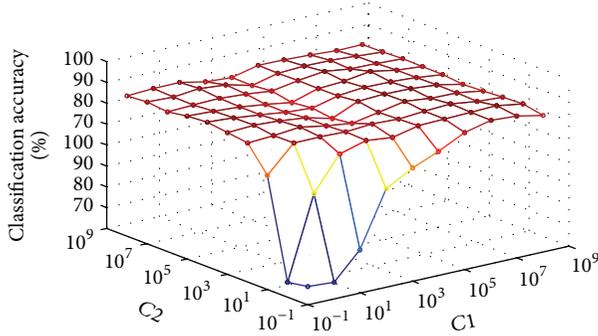
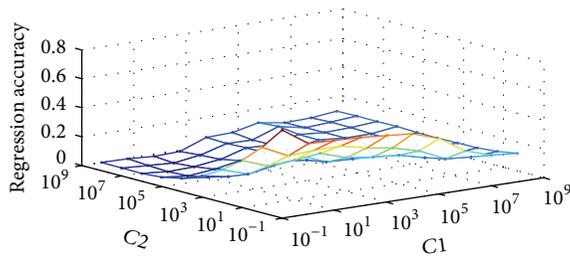
(1) *Why is the learning speed of DBMK-ELM much slower than basic ELM in some cases?* The main reason is that there are substantial samples to learn in the  $K$ -space, which is constructed in multiple kernel learning step. Specifically, if there are  $n$  original samples, there will be  $n^2$  corresponding new samples. Therefore, the training time difference between DBMK-ELM and basic ELM will be magnified with the training samples increasing. It may be possible to reduce the

training time gap between DBMK-ELM and basic ELM if we use sampling techniques in the  $K$ -space.

(2) *In which cases should we use DBMK-ELM?* DBMK-ELM can obtain more accurate results and a faster learning speed compared with traditional multiple kernel learning method, SimpleMKL. Despite the fact that it is much better than other multiple kernel learning methods, DBMK-ELM has more time cost compared with basic ELM method. In this way, a trade-off between accuracy and time cost is needed. The experimental results show that DBMK-ELM significantly improves testing accuracy compared with basic ELM in regression and multisource data fusion problems in most cases. But in the classification case, where kernels are generated from one data source, DBMK-ELM has no significant difference compared with basic ELM in testing accuracy. Therefore, a preferable choice is to apply DBMK-ELM in regression and multisource data fusion problems and use basic ELM in single data source generated kernel classification problems.

TABLE 9: Multiple kernel classification case: classification training time (s).

| Data     | DBMK-ELM            | SimpleMKL [16]       | $\ell_1$ -MK-ELM [23] | R-MK-ELM [23]        | ELM [8]             | UW                  |
|----------|---------------------|----------------------|-----------------------|----------------------|---------------------|---------------------|
| Plant    | 0.6545 $\pm$ 0.0272 | 9.1479 $\pm$ 0.6296  | 4.5658 $\pm$ 0.5378   | 7.8680 $\pm$ 0.6125  | 0.0064 $\pm$ 0.0006 | 0.0065 $\pm$ 0.0006 |
| PsortPos | 0.2062 $\pm$ 0.0124 | 2.3172 $\pm$ 0.2031  | 0.8097 $\pm$ 0.0893   | 2.4892 $\pm$ 0.3435  | 0.0019 $\pm$ 0.0005 | 0.0017 $\pm$ 0.0003 |
| PsortNeg | 1.5343 $\pm$ 0.1060 | 25.5086 $\pm$ 1.1290 | 11.9782 $\pm$ 0.7766  | 24.7445 $\pm$ 1.0424 | 0.0187 $\pm$ 0.0009 | 0.0187 $\pm$ 0.0011 |
| AVG      | 0.7983              | 12.3246              | 5.7846                | 11.7006              | 0.0090              | 0.0090              |

FIGURE 2: Classification case: performances of DBMK-ELM with different parameters on the *ionosphere* data set.FIGURE 3: Regression case: performances of DBMK-ELM with different parameters on the *yacht* data set.

## 5. Conclusion

In this paper, we have proposed DBMK-ELM, a new multiple kernel based ELM, to extend the basic kernel based ELM. The proposed multiple kernel learning method can unify classification and regression problems. Moreover, DBMK-ELM is able to learn from multiple kernels at an extremely fast speed. Experimental results show that DBMK-ELM achieves a significant performance enhancement, in terms of the accuracy and the time cost in both classification and regression problems. In future, we will consider how to define a better distance among different classes and how to extend DBMK-ELM to the semisupervised learning problem.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Project nos. 60970034, 61170287, and 61232016), the National Basic Research Program of China (973) under Grant no. 2014CB340303, and the Hunan Provincial Science and Technology Planning Project of China (Project no. 2012FJ4269).

## References

- [1] J. Yin, E. Zhu, X. Yang, G. Zhang, and C. Hu, "Two steps for fingerprint segmentation," *Image and Vision Computing*, vol. 25, no. 9, pp. 1391–1403, 2007.
- [2] C. Zhu, J. Yin, and Q. Li, "A stock decision support system based on DBNs," *Journal of Computational Information Systems*, vol. 10, no. 2, pp. 883–893, 2014.
- [3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [4] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 2000.
- [5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, July 2004.
- [6] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [7] Y. Wang, F. Cao, and Y. Yuan, "A study on effectiveness of extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2483–2490, 2011.
- [8] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [9] Y. Yuan, Y. Wang, and F. Cao, "Optimization approximation solution for regression problem based on extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2475–2482, 2011.
- [10] J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, and V. Leung, "A secure and practical mechanism of outsourcing extreme learning machine in cloud computing," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 35–38, 2013.
- [11] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [12] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.

- [13] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.
- [14] Z. Bai, G. B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse extreme learning machine for classification," *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1858–1870, 2014.
- [15] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.
- [16] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research (JMLR)*, vol. 9, pp. 2491–2521, 2008.
- [17] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, " $L_p$ -norm multiple kernel learning," *The Journal of Machine Learning Research*, vol. 12, pp. 953–997, 2011.
- [18] X. Liu, L. Wang, J. Yin, and L. Liu, "Incorporation of radius-info can be simple with SimpleMKL," *Neurocomputing*, vol. 89, pp. 30–38, 2012.
- [19] X. Liu, L. Wang, J. Zhang, and J. Yin, "Sample-adaptive multiple kernel learning," AAAI, 2014.
- [20] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *Advances in Neural Information Processing Systems*, vol. 14, pp. 367–373, MIT Press, 2002, <http://papers.nips.cc/paper/1946-on-kernel-target-alignment.pdf>.
- [21] A. Kumar, A. Niculescu-Mizil, K. Kavukcoglu, and H. Daumé, "A binary classification framework for two-stage multiple kernel learning," in *Proceedings of the 29th International Conference on Machine Learning (ICML '12)*, pp. 1295–1302, July 2012.
- [22] L.-J. Su and M. Yao, "Extreme learning machine with multiple kernels," in *Proceedings of the 10th IEEE International Conference on Control and Automation (ICCA '13)*, pp. 424–429, June 2013.
- [23] X. Liu, L. Wang, G. B. Huang, J. Zhang, and J. Yin, "Multiple kernel extreme learning machine," *Neurocomputing*, vol. 149, part A, pp. 253–264, 2015.
- [24] K. Bache and M. Lichman, UCI machine learning repository, 2013, <http://archive.ics.uci.edu/ml>.
- [25] <http://lib.stat.cmu.edu/datasets/>.
- [26] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne, "Predicting subcellular localization of proteins based on their N-terminal amino acid sequence," *Journal of Molecular Biology*, vol. 300, no. 4, pp. 1005–1016, 2000.
- [27] J. L. Gardy, M. R. Laird, F. Chen et al., "PSORTb v.2.0: expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis," *Bioinformatics*, vol. 21, no. 5, pp. 617–623, 2005.

## Research Article

# Deep Extreme Learning Machine and Its Application in EEG Classification

Shifei Ding,<sup>1,2</sup> Nan Zhang,<sup>1,2</sup> Xinzheng Xu,<sup>1,2</sup> Lili Guo,<sup>1,2</sup> and Jian Zhang<sup>1,2</sup>

<sup>1</sup>School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

<sup>2</sup>Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

Correspondence should be addressed to Shifei Ding; [dingsf@cumt.edu.cn](mailto:dingsf@cumt.edu.cn)

Received 26 August 2014; Revised 4 November 2014; Accepted 12 November 2014

Academic Editor: Amaury Lendasse

Copyright © 2015 Shifei Ding et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, deep learning has aroused wide interest in machine learning fields. Deep learning is a multilayer perceptron artificial neural network algorithm. Deep learning has the advantage of approximating the complicated function and alleviating the optimization difficulty associated with deep models. Multilayer extreme learning machine (MLELM) is a learning algorithm of an artificial neural network which takes advantages of deep learning and extreme learning machine. Not only does MLELM approximate the complicated function but it also does not need to iterate during the training process. We combining with MLELM and extreme learning machine with kernel (KELM) put forward deep extreme learning machine (DELIM) and apply it to EEG classification in this paper. This paper focuses on the application of DELIM in the classification of the visual feedback experiment, using MATLAB and the second brain-computer interface (BCI) competition datasets. By simulating and analyzing the results of the experiments, effectiveness of the application of DELIM in EEG classification is confirmed.

## 1. Introduction

Brain-computer interface (BCI) is a kind of technology that enables people to communicate with a computer or to control devices with EEG signals [1]. The core technologies of BCI are to extract the feature of preprocessed EEG and classify ready-processed EEG, and this paper is mainly about classification analysis. In recent years, BCI has gotten a great advance with the rapid development of computer technology. BCI has been applied to many fields, such as medicine and military [2–4]. Currently, many different methods have been proposed for EEG classification, including decision trees, local backpropagation (BP) algorithm, Bayes classifier,  $K$ -nearest neighbors (KNN), support vector machine (SVM), batch incremental support vector machine (BISVM), and ELM [5–8]. However, most of them are shallow neural network algorithms in which the capabilities achieve approximating the complex functions that are subject to certain restrictions, and there is no such restriction in deep learning.

Deep learning is an artificial neural network learning algorithm which has multilayer perceptrons. Deep learning

has achieved an approximation of complex functions and alleviated the optimization difficulty associated with the deep models [9–11]. In 2006, the concept of deep learning was first proposed by Hinton and Salakhutdinov who presented deep structure of multilayer autoencoder [12]. Deep belief network was proposed by Hinton [13]. LeCun et al. put forward the first real deep learning algorithm—convolutional neural networks (CNNs) [14]. More and more people put forward some new algorithms based on deep learning. Then convolutional deep belief network was put forward [15]. In 2013, the model of multilayer extreme learning machine (MLELM) was proposed by Kasun et al. [16], and DELIM takes advantages of deep learning and extreme learning machine. Extreme learning machine (ELM) proposed by Huang et al. is a simple and efficient learning algorithm of single layer feed-forward neural networks (SLFNs) [17, 18]. In addition, some people put forward some deformation algorithms based on ELM, such as regularized extreme learning machine (RELM) [19], extreme learning machine with kernel (KELM) [20], optimally pruned extreme learning machine (OP-ELM) [21], and evolving fuzzy optimally pruned extreme learning machine (eF-OP-ELM) [22].

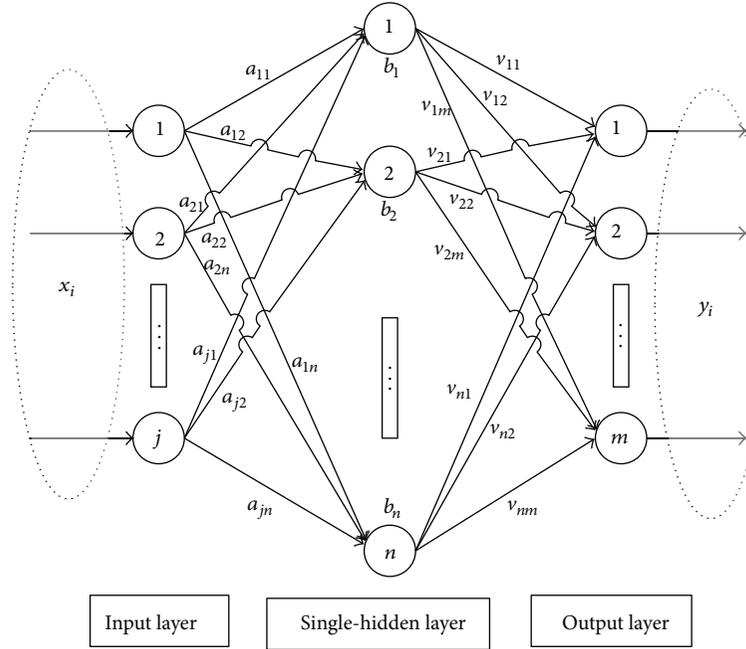


FIGURE 1: The model structure of ELM.

We combining with multilayer extreme learning machine (MLELM) and extreme learning machine with kernel (KELM) put forward deep extreme learning machine (DELIM) and apply it to EEG classification, and the paper is organized as follows: Section 2 gives the model of ELM, RELM, and KELM. Section 3 describes the model structure of MLELM. Section 4 details the model structure of DELIM. Section 5 first evaluates the usefulness of DELIM on UCI datasets and then applies DELIM to EEG classification. In Section 6, the conclusion is gotten.

## 2. Extreme Learning Machine (ELM)

**2.1. Basic Extreme Learning Machine (Basic ELM).** ELM proposed by Huang et al. is a simple and efficient learning algorithm of SLFNs. The model of ELM constituted input layer, single-hidden layer, and output layer. The model structure of ELM is shown in Figure 1, with  $j$  input layer nodes,  $n$  hidden layer nodes,  $m$  output layer nodes, and the hidden layer activation function  $g(x)$ .

For  $N$  distinct samples  $x_i \in R_N \times R_j$ ,  $y_i \in R_N \times R_m$  ( $i = 1, 2, \dots, N$ ), the outputs of the hidden layer can be expressed as (1), and the numerical relationship between output of the hidden layer and output of the output layer can be expressed as (2):

$$h = g(ax + b), \quad (1)$$

$$h(x_i)V = y_i, \quad i = 1, 2, \dots, N. \quad (2)$$

The above equation can be written compactly as

$$HV = Y, \quad (3)$$

where

$$H = \begin{bmatrix} g(\vec{a}_1, b_1, \vec{x}_1) & g(\vec{a}_1, b_1, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \\ g(\vec{a}_2, b_2, \vec{x}_1) & g(\vec{a}_2, b_2, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ g(\vec{a}_n, b_n, \vec{x}_1) & g(\vec{a}_n, b_n, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \end{bmatrix}^T, \quad (4)$$

$$V = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}_{n \times m}, \quad Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times m}, \quad (5)$$

where  $a_i = [a_{i1}, a_{i2}, \dots, a_{in}]^T$  are the weights connecting the  $i$ th input nodes and hidden layer,  $b_j$  is the bias of the  $j$ th hidden node, and  $v_j = [v_{j1}, v_{j2}, \dots, v_{jm}]^T$  are the weights connecting the  $j$ th hidden node and the output layer.  $H$  is output matrix of the neural network. We need to set input weights  $a_{ij}$  and the bias of the hidden layer  $b_j$ ; the output weights  $V$  can be obtained by a series of linear equations transformations.

In conclusion, using ELM to obtain the output weights  $V$  can be divided into three steps.

*Step 1.* Randomly select numerical values between 0 and 1 to set input weights  $a_{ij}$  and the bias of the hidden layer  $b_j$ .

*Step 2.* Calculate the output matrix  $H$ .

*Step 3.* Calculate the output weights  $V$ :

$$V = H^\dagger Y, \quad (6)$$

where  $H^\dagger$  represents the generalized inverse matrix of the output matrix  $H$ .

2.2. *Regularized Extreme Learning Machine (RELM)*. ELM has the advantage of fast training speed and high generalization performance, but ELM also has the disadvantage of bad robustness. Deng et al. combining with experiential risk and structural risk put forward regularized extreme learning machine (RELM) which has better robustness, and RELM aims to solve the output weights by minimizing the regularized cost function of least squares estimate regularization, which leads to the following formulation:

$$\min L_{\text{RELM}} = \frac{1}{2} \|V\|^2 + \frac{C}{2} \|Y - HV\|^2, \quad (7)$$

where  $C$  is a scale parameter which adjusts experiential risk and structural risk.

By setting the gradient of  $L_{\text{RELM}}$  with respect to  $\mathbf{V}$  to zero, we have

$$V + CH^T(Y - HV) = 0. \quad (8)$$

When the number of training samples is more than the number of hidden layer nodes, the output weight matrix  $\mathbf{V}$  in RELM can be expressed as

$$V = \left( \frac{I}{C} + H^T H \right)^{-1} H^T Y. \quad (9)$$

When the number of training samples is less than the number of hidden layer nodes, the output weight matrix  $\mathbf{V}$  in RELM can be expressed as

$$V = H^T \left( \frac{I}{C} + HH^T \right)^{-1} Y. \quad (10)$$

2.3. *Extreme Learning Machine with Kernel (KELM)*. Huang et al. combining with the kernel method and extreme learning machine put forward extreme learning machine with kernel (KELM). The outputs of the hidden layer of ELM can be regarded as the nonlinear mapping of samples. When the mapping is an unknown, we can construct the kernel function instead of  $HH^T$ :

$$HH^T = \Omega_{\text{ELM}} = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_N) \\ \vdots & \ddots & \vdots \\ K(x_N, x_1) & \cdots & K(x_N, x_N) \end{bmatrix}, \quad (11)$$

$$h(x)H^T = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T.$$

The most popular kernel of KELM in use is the Gaussian kernel  $K(x_i, x_j) = \exp(-\|x_i - x_j\|/\gamma)$ , where  $\gamma$  is the kernel parameter.

Thus, the output weight matrix  $\mathbf{V}$  in KELM can be expressed as (12) and the Classification of formula of KELM can be expressed as (13):

$$V = \left( \frac{I}{C} + \Omega_{\text{ELM}} \right)^{-1} Y, \quad (12)$$

$$f(x) = h(x)H^T V = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T \left( \frac{I}{C} + \Omega_{\text{ELM}} \right)^{-1} Y. \quad (13)$$

### 3. Multilayer Extreme Learning Machine (MLELM)

3.1. *Extreme Learning Machine-Autoencoder (ELM-AE)*. Autoencoder is an artificial neural network model which is commonly used in deep learning. Autoencoder is an unsupervised neural network, the outputs of autoencoder are the same as the inputs of autoencoder, and autoencoder is a kind of neural networks which reproduces the input signal as much as possible. ELM-AE proposed by Kasun et al. is a new method of neural network which can reproduce the input signal as well as autoencoder.

The model of ELM-AE constituted input layer, single-hidden layer, and output layer. The model structure of ELM-AE is shown in Figure 2, with  $j$  input layer nodes,  $n$  hidden layer nodes,  $j$  output layer nodes, and the hidden layer activation function  $g(x)$ . According to the output of the hidden layer representing the input signal, ELM-AE can be divided into three different representations as follows.

$j > n$ : Compressed Representation: this represents features from a higher dimensional input signal space to a lower dimensional feature space.

$j = n$ : Equal Dimension Representation: this represents features from an input signal space dimension equal to feature space dimension.

$j < n$ : Sparse Representation: this represents features from a lower dimensional input signal space to a higher dimensional feature space.

There are two differences between ELM-AE and traditional ELM. Firstly, ELM is a supervised neural network and the output of ELM is label, but ELM-AE is an unsupervised neural network and the output of ELM-AE is the same as the input of ELM-AE. Secondly, the input weights of ELM-AE are orthogonal and the bias of hidden layer of ELM-AE is also orthogonal, but ELM is not so. For  $N$  distinct samples,  $x_i \in R_n \times R_j$ , ( $i = 1, 2, \dots, N$ ), the outputs of ELM-AE hidden layer can be expressed as (14), and the numerical relationship between the outputs of the hidden layer and the outputs of the output layer can be expressed as (15):

$$h = g(ax + b), \quad \text{where } a^T a = I, b^T b = 1, \quad (14)$$

$$h(x_i)V = x_i^T, \quad i = 1, 2, \dots, N. \quad (15)$$

Using ELM-AE to obtain the output weights  $V$  can be also divided into three steps, but the calculation method of the output weights  $V$  of ELM-AE in Step 3 is different from the calculation method of the output weights  $V$  of ELM.

For sparse and compressed ELM-AE representations, output weights  $\mathbf{V}$  are calculated by (16) and (17).

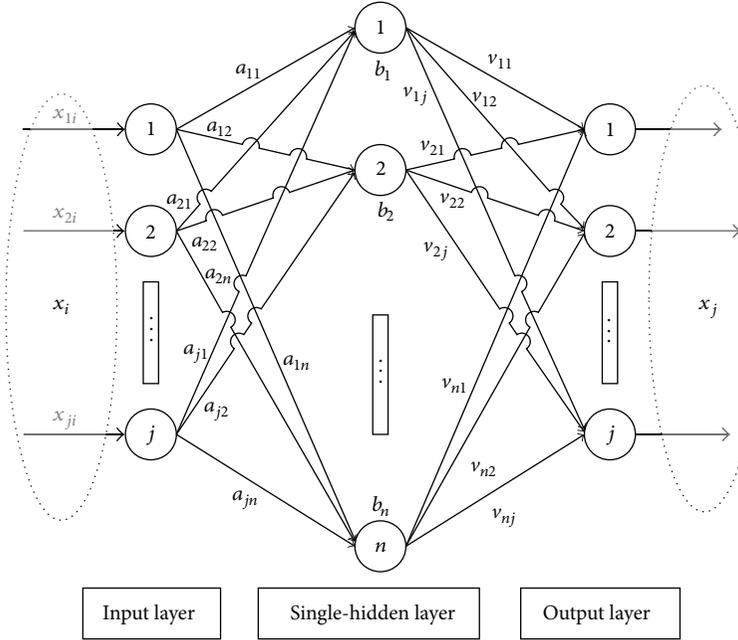


FIGURE 2: The model structure of ELM-AE.

When the number of training samples is more than the number of hidden layer nodes,

$$V = \left( \frac{I}{C} + H^T H \right)^{-1} H^T X. \quad (16)$$

When the number of training samples is less than the number of hidden layer nodes,

$$V = H^T \left( \frac{I}{C} + H H^T \right)^{-1} X. \quad (17)$$

For equal dimension ELM-AE representation, output weights  $V$  are calculated by

$$V = H^{-1} X. \quad (18)$$

**3.2. Multilayer Extreme Learning Machine (MLELM).** In 2006, Hinton et al. put forward an effective method of establishing a multilayer neural network on the unsupervised data. In the new method, first the parameters in each layer are obtained by unsupervised training, and then the network is fine-tuned by supervised learning. In 2013, MLELM was proposed by Kasun et al. Like other deep learning models, MLELM makes use of unsupervised learning to train the parameters in each layer, but the difference is that MLELM does not need to fine-tune the network. Thus, compared with other deep learning algorithms, MLELM does not need to spend a long time on the network training.

MLELM makes use of ELM-AE to train the parameters in each layer, and MLELM hidden layer activation functions can be either linear or nonlinear piecewise. If the activation function of the MLELM  $i$ th hidden layer is  $g(x)$ , then the

parameters between the MLELM  $i$ th hidden layer and the MLELM  $(i-1)$  hidden layer (if  $i-1=0$ , this layer is the input layer) are trained by ELM-AE, and the activation function should be  $g(x)$ , too. The numerical relationship between the outputs of MLELM  $i$ th hidden layer and the outputs of MLELM  $(i-1)$  hidden layer can be expressed as

$$H_i = g\left((v_i)^T H_{i-1}\right), \quad (19)$$

where  $H_i$  represents the outputs of MLELM  $i$ th hidden layer (if  $i-1=0$ , this layer is the input layer, and  $H_{i-1}$  represents the inputs of MLELM). The model of MLELM is shown in Figure 3,  $v_i$  represents the output weights of ELM-AE, the input of ELM-AE is  $H_{i-1}$ , and the number of ELM-AE hidden layer nodes is identical to the number of MLELM  $i$ th hidden nodes when the parameters between the MLELM  $i$ th hidden layer and the MLELM  $(i-1)$  hidden layer are trained by ELM-AE. The output of the connections between the last hidden layer and the output layer can be analytically calculated using regularized least squares.

#### 4. Deep Extreme Learning Machine (DELM)

MLELM makes use of ELM-AE to train the parameters in each layer, and ML-ELM hidden layer activation functions can be either linear or nonlinear piecewise, and the mapping of MLELM is linear or nonlinear. When the mapping is an unknown, we can add one hidden layer and construct the kernel function. In other words, at last the outputs of MLELM hidden layer  $H_k$  (the matrix size is  $n_k * N$ ) are the inputs of KELM, and we can construct the kernel function instead

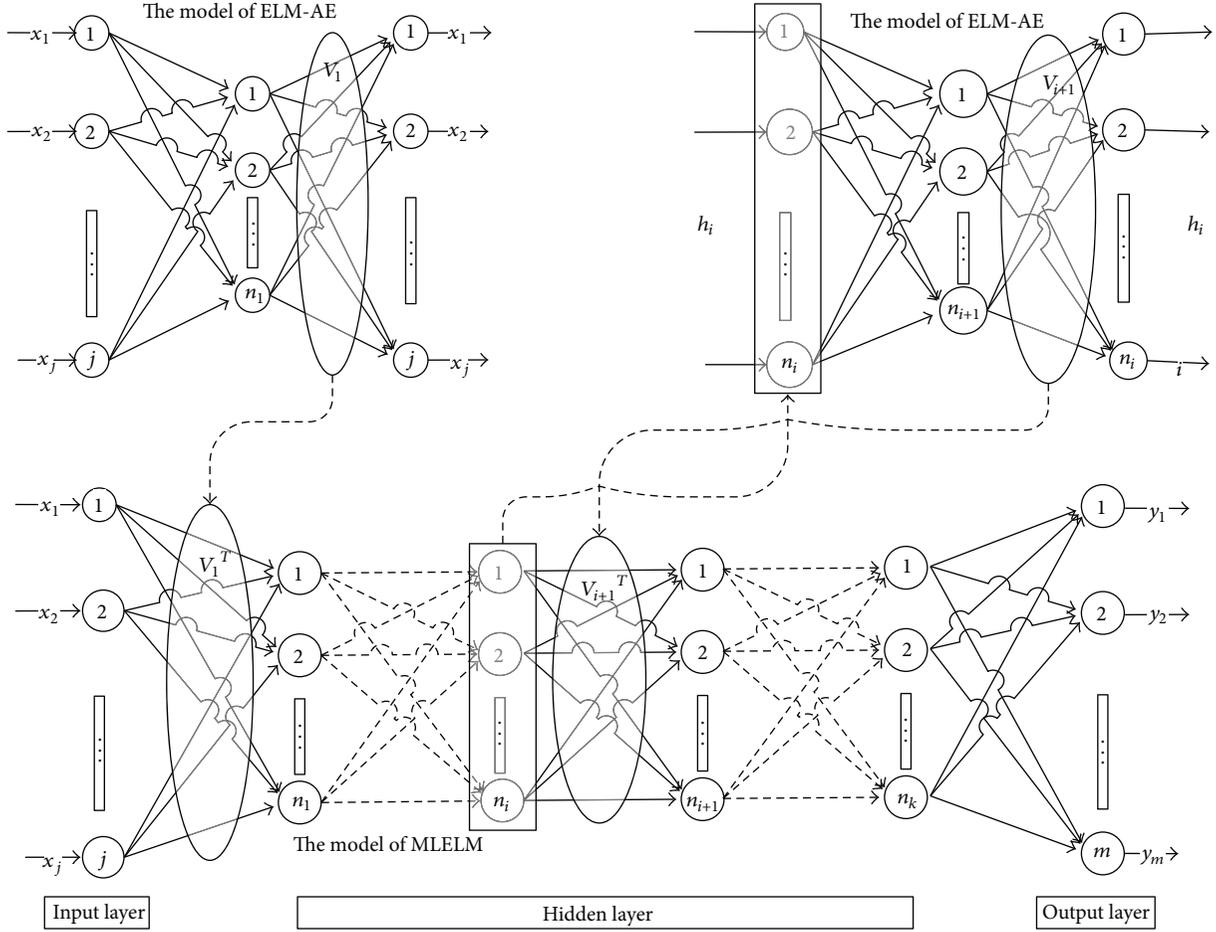


FIGURE 3: The model structure of MLELM.

of  $H_{k+1}H_{k+1}^T$ . This algorithm combining with MLELM and KELM is called deep extreme learning machine (DELM):

$$\begin{aligned}
 & H_{k+1}H_{k+1}^T \\
 &= \Omega_{\text{DELM}} \\
 &= \begin{bmatrix} K(H_k(:, 1), H_k(:, 1)) & \cdots & K(H_k(:, 1), H_k(:, N)) \\ \vdots & \ddots & \vdots \\ K(H_k(:, N), H_k(:, 1)) & \cdots & K(H_k(:, N), H_k(:, N)) \end{bmatrix}, \\
 & h_{k+1}(h_k(x))H_{k+1}^T = \begin{bmatrix} K(h_k(x), H_k(:, 1)) \\ \vdots \\ K(h_k(x), H_k(:, N)) \end{bmatrix}^T.
 \end{aligned} \tag{20}$$

The model of DELM is shown in Figure 4,  $v_i$  ( $i \in [1, \dots, k]$ ) represents the output weights of ELM-AE, the input of ELM-AE is  $H_{i-1}$ , and the number of ELM-AE hidden layer nodes is identical to the number of DELM  $i$ th hidden nodes when the parameters between the DELM  $i$ th hidden layer and the MLELM  $(i - 1)$  hidden layer are trained by ELM-AE. And we can construct the kernel function instead of  $H_{k+1}H_{k+1}^T$ ; thus the output weight matrix  $V$  in DELM can

be expressed as (21) and the classification formula of KELM can be expressed as (22):

$$V = \left( \frac{I}{C} + \Omega_{\text{DELM}} \right)^{-1} Y, \tag{21}$$

$$\begin{aligned}
 f(x) &= h_{k+1}(h_k(x))H_{k+1}^T V \\
 &= \begin{bmatrix} K(h_k(x), H_k(:, 1)) \\ \vdots \\ K(h_k(x), H_k(:, N)) \end{bmatrix}^T \left( \frac{I}{C} + \Omega_{\text{DELM}} \right)^{-1} Y.
 \end{aligned} \tag{22}$$

## 5. Experiments and Analysis

The execution environment of experiments is MATLAB 2012B. All activation functions of ELM, MLELM, and DELM select sigmoid function and the kernel functions of KELM and DELM are Gaussian kernel. ELM, MLELM, and DELM were executed 100 times, and the average values and the best values are reported.

**5.1. UCI Datasets Classification.** In this part, the UCI datasets were used to test the performances of DELM, and the details

TABLE 1: The details of UCI datasets.

| Dataset    | Number of samples |                 | Attributes information |                           | Number of labels |
|------------|-------------------|-----------------|------------------------|---------------------------|------------------|
|            | Training samples  | Testing samples | Number of attributes   | Attribute characteristics |                  |
| Ionosphere | 200               | 151             | 34                     | Continuous attributes     | 2                |
| Diabetes   | 576               | 192             | 8                      | Categorical, integer      | 2                |

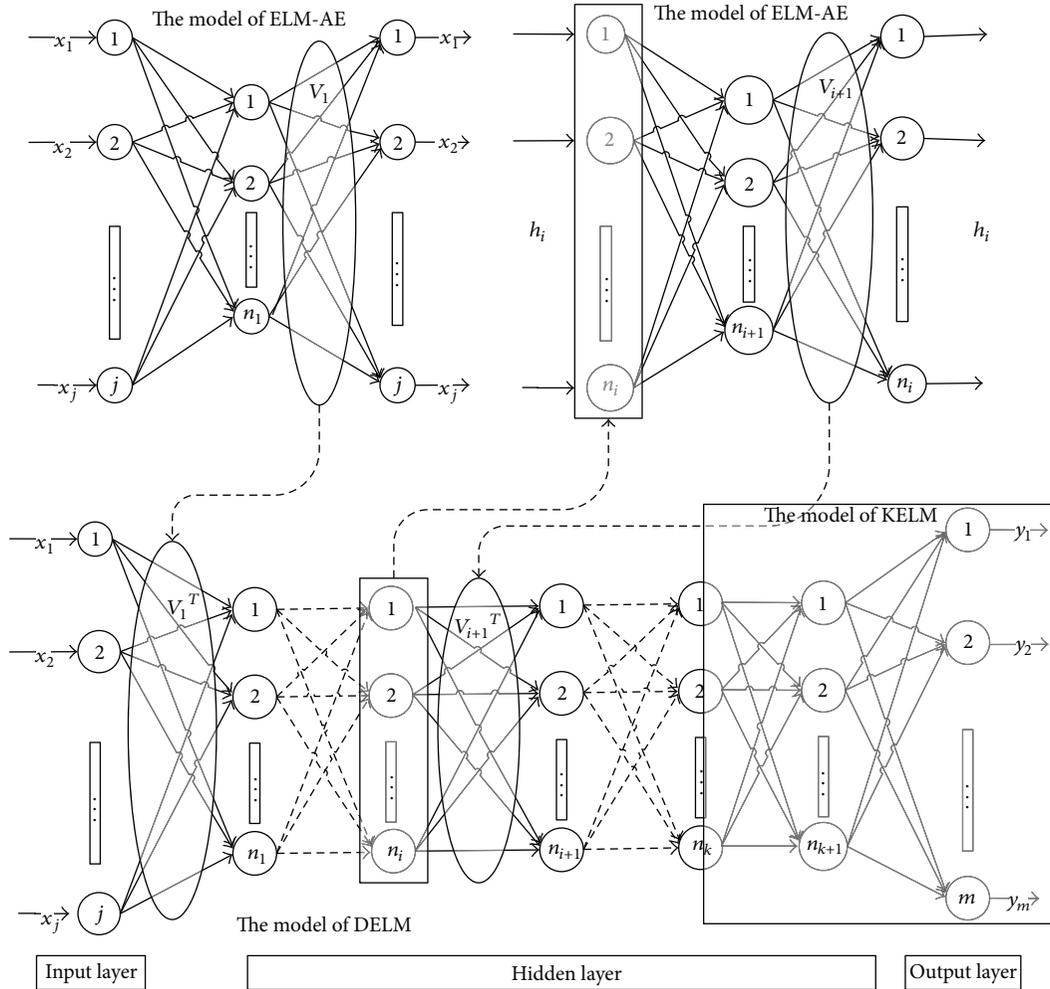


FIGURE 4: The model structure of DELM.

of UCI dataset are presented in Table 1, including ionosphere dataset and diabetes dataset.

As shown in Figure 5, we can make choices that the numbers of ELM hidden layer nodes on ionosphere dataset and diabetes dataset are 50 and 40, the regularized parameter  $C$  and the kernel parameter  $\gamma$  of KELM on ionosphere dataset are  $10^3$  and  $10^2$ , and the regularized parameter  $C$  and the kernel parameter  $\gamma$  of KELM on diabetes dataset are  $10^2$  and  $10^1$ . The structure of MLELM on ionosphere dataset is 34-30-30-50-2, where the parameter  $C$  for layer 34-30 is  $10^3$ , the parameter  $C$  for layer 30-50 is  $10^{-2}$ , and the parameter  $C$  for layer 50-2 is  $10^8$ . And the structure of MLELM on diabetes dataset is 8-10-10-40-2, where the parameter  $C$  for layer 8-10 is

$10^6$ , the parameter  $C$  for layer 10-40 is  $10^8$ , and the parameter  $C$  for layer 40-2 is  $10^5$ . The structure of DELM on ionosphere dataset is 34-30-30-L-2, where the parameter  $C$  for layer 34-30 is  $10^1$ , the parameter  $C$  for layer L-2 is  $10^3$ , and the kernel parameter  $\gamma$  is  $10^2$ . And the structure of DELM on diabetes dataset is 8-10-10-L-2, where the parameter  $C$  for layer 34-30 is  $10^1$ , the parameter  $C$  for layer L-2 is  $10^2$ , and the kernel parameter  $\gamma$  is  $10^1$ .

The performance comparison of DELM with ELM, KELM, and MLELM on UCI datasets is shown in Table 2. It is clearly observed that DELM testing accuracy is higher than MLELM, either the average or the maximum, and the best values of DELM testing accuracy are higher than ELM and

TABLE 2: Performance comparison of DELM with ELM, KELM, and MLELM on UCI datasets.

| Dataset    | Algorithm | #       | Training accuracy   | Testing accuracy    | Training time (s)   | Testing time (s)    |
|------------|-----------|---------|---------------------|---------------------|---------------------|---------------------|
| Ionosphere | Basic ELM | Average | $0.9207 \pm 0.0151$ | $0.9342 \pm 0.0222$ | $0.0044 \pm 0.0074$ | $0.0013 \pm 0.0048$ |
|            |           | Best    | 0.9500              | 0.9735              | —                   | —                   |
|            | KELM      | —       | 0.9900              | 0.9735              | 0.0064              | 0.0017              |
|            |           | ML-ELM  | Average             | $0.9112 \pm 0.0159$ | $0.9447 \pm 0.0216$ | $0.0115 \pm 0.0116$ |
|            | DELM      | Best    | 0.9500              | 0.9801              | —                   | —                   |
|            |           | Average | $0.9503 \pm 0.0111$ | $0.9474 \pm 0.0292$ | $0.0164 \pm 0.0079$ | $0.0045 \pm 0.0064$ |
| Diabetes   | Basic ELM | Average | $0.7983 \pm 0.0062$ | $0.7725 \pm 0.0129$ | $0.0072 \pm 0.0132$ | $0.0034 \pm 0.0098$ |
|            |           | Best    | 0.8125              | 0.8021              | —                   | —                   |
|            | KELM      | —       | 0.7899              | 0.7917              | 0.6818              | 0.0314              |
|            |           | ML-ELM  | Average             | $0.7666 \pm 0.0207$ | $0.7522 \pm 0.0324$ | $0.0091 \pm 0.0143$ |
|            | DELM      | Best    | 0.7951              | 0.8177              | —                   | —                   |
|            |           | Average | $0.7871 \pm 0.0079$ | $0.7580 \pm 0.0422$ | $0.0641 \pm 0.0143$ | $0.0112 \pm 0.0086$ |
| DELM       | Best      | 0.8038  | 0.8229              | —                   | —                   |                     |

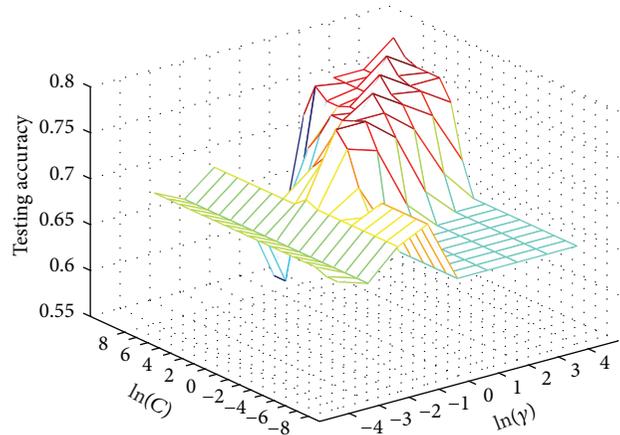
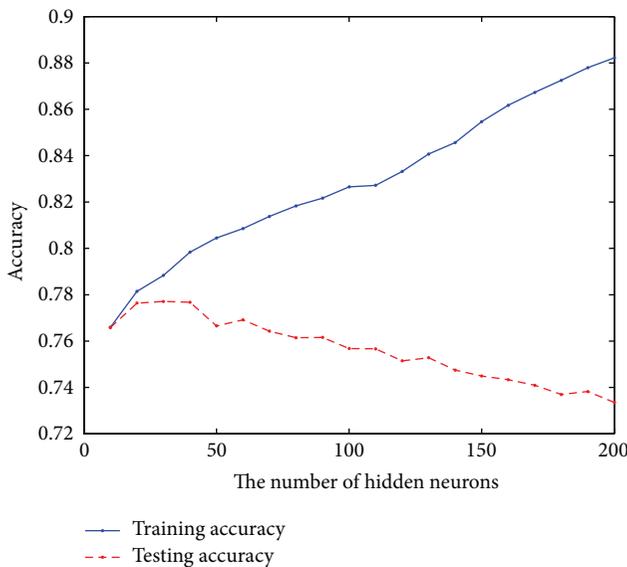
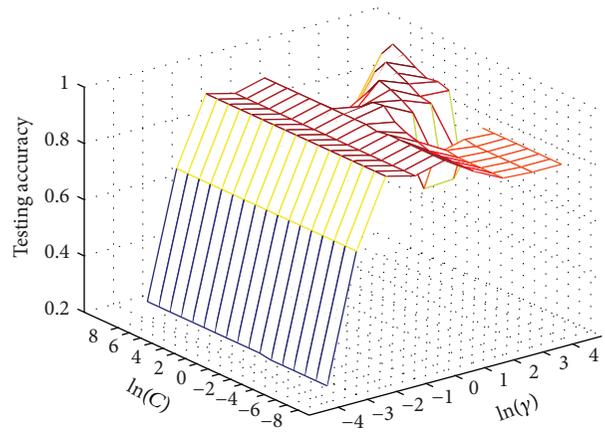
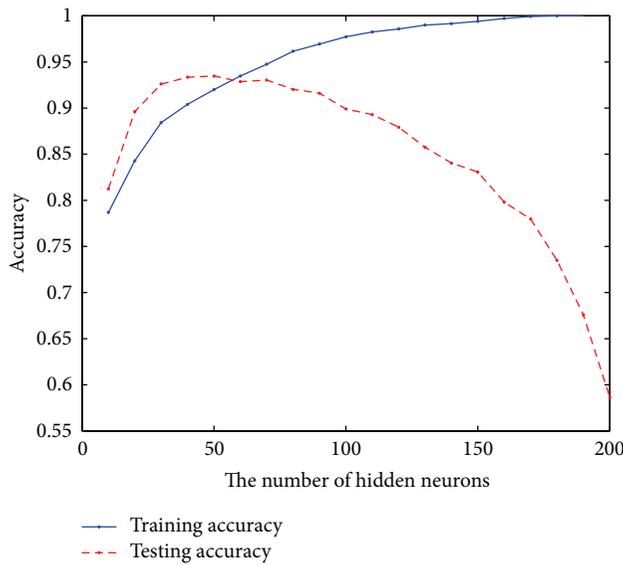


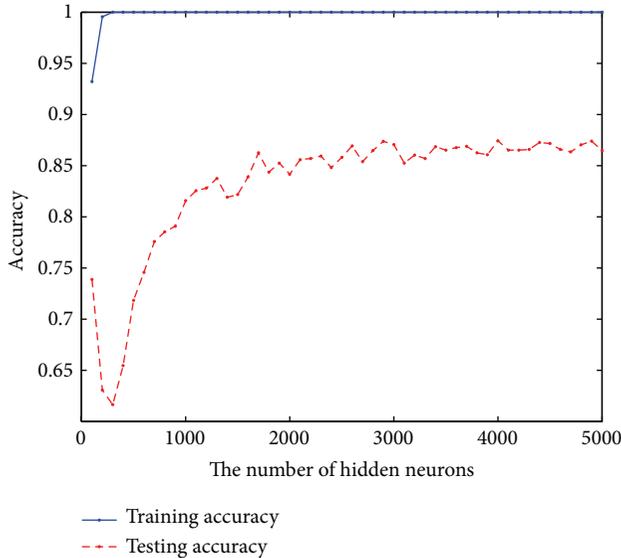
FIGURE 5: Basic ELM and KELM for UCI dataset.

TABLE 3: The details of the second BCI competition dataset IA.

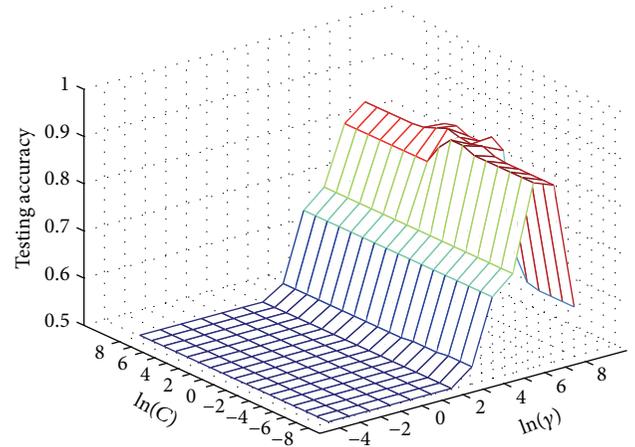
| Dataset                       | Number of samples |                 | Number of attributes | Number of labels |
|-------------------------------|-------------------|-----------------|----------------------|------------------|
|                               | Training samples  | Testing samples |                      |                  |
| BCI competition II dataset IA | 268               | 293             | 5376                 | 2                |

TABLE 4: Performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IA.

| Dataset                       | Algorithm | #       | Training accuracy   | Testing accuracy    | Training time (s)   | Testing time (s)    |
|-------------------------------|-----------|---------|---------------------|---------------------|---------------------|---------------------|
| BCI competition II dataset IA | Basic ELM | Average | $1.0000 \pm 0$      | $0.8609 \pm 0.0187$ | $3.3670 \pm 0.0866$ | $2.2361 \pm 0.0498$ |
|                               |           | Best    | 1.0000              | 0.9078              | —                   | —                   |
|                               | KELM      | —       | 0.8582              | 0.9010              | 0.0754              | 0.1430              |
|                               | ML-ELM    | Average | $0.7849 \pm 0.0213$ | $0.8642 \pm 0.0216$ | $9.2012 \pm 0.1444$ | $0.4820 \pm 0.0272$ |
|                               |           | Best    | 0.8358              | 0.9113              | —                   | —                   |
|                               | DELM      | Average | $0.7515 \pm 0.0161$ | $0.8650 \pm 0.0224$ | $6.7438 \pm 0.2099$ | $0.2932 \pm 0.0290$ |
| Best                          |           | 0.7873  | 0.9181              | —                   | —                   |                     |



(a) Basic ELM for the BCI competition II dataset IA



(b) KELM for the BCI competition II dataset IA

FIGURE 6: Basic ELM and KELM for the BCI competition II dataset IA.

KELM. And DELM training time is the longest, but there is little difference between testing times. Sigillito et al. investigated ionosphere dataset using backpropagation and the perceptron training algorithm; they found that “linear” perceptron achieved 90.7%, a “nonlinear” perceptron achieved 92%, and backprop an average of over 96% accuracy [23]. Although the average value of DELM on ionosphere dataset only achieves 94.74%, the best value has reached to 99.34%.

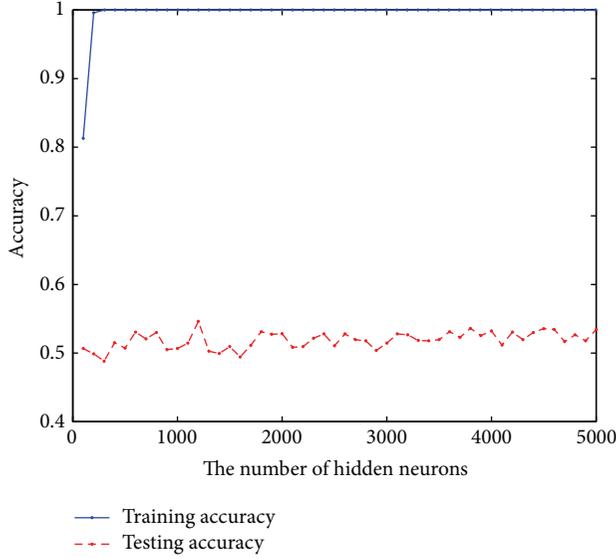
**5.2. EEG Classification.** The effectiveness of DELM has been confirmed, so the effectiveness of the application of DELM in EEG classification is tested in this part.

**5.2.1. Visual Feedback Experiment (Healthy Subject).** The performances of DELM on the second BCI competition dataset IA are tested in this section, and this dataset comes from

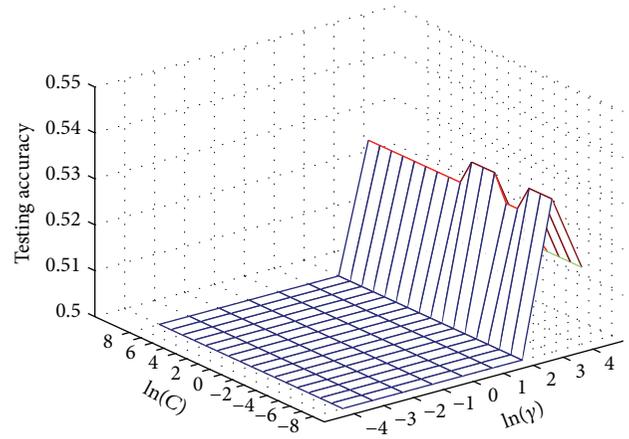
the visual feedback experiment (healthy subject) provided by University of Tuebingen [24].

The datasets were taken from a healthy subject. The subject was asked to move a cursor up and down on a computer screen, while his cortical potentials were taken. Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor. Each trial lasted 6 s. The visual feedback was presented from second 2 to second 5.5. Only this 3.5-second interval of every trial is provided for training and testing. The sampling rate of 256 Hz and the recording length of 3.5 s result in 896 samples per channel for every trial, and the details are presented in Table 3.

As shown in Figure 6, we can make choices that the number of ELM hidden layer nodes on BCI competition II dataset IA is 3000; the regularized parameter  $C$  and the kernel parameter  $\gamma$  of KELM are  $10^3$  and  $10^4$ . The structure of



(a) Basic ELM for the BCI competition II dataset IA



(b) KELM for the BCI competition II dataset IA

FIGURE 7: Basic ELM and KELM for the BCI competition II dataset IA.

MLELM is 5376-500-500-3000-2, where the parameter  $C$  for layer 5376-500 is  $2^1$ , the parameter  $C$  for layer 500-3000 is  $2^8$ , and the parameter  $C$  for layer 3000-2 is  $2^{-7}$ . The structure of DELM is 5376-500-500-L-2, where the parameter  $C$  for layer 5376-500 is  $10^{-1}$ , the parameter  $C$  for layer L-2 is  $10^{-1}$ , and the kernel parameter  $\gamma$  is  $10^2$ .

The performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IA is shown in Table 4. It is clearly observed that DELM testing accuracy is higher than MLELM, either the average or the maximum, and the best values of DELM testing accuracy are higher than ELM and KELM. MLELM training time is the longest, and the testing time of MLELM and DELM is less than ELM. The performance comparison of DELM with the results of BCI competition II dataset IA is shown in Table 5. It is clear that the average error value of DELM on BCI competition II dataset IA achieves 13.50%, but the min error value has reduced to 8.19%, which is much lower than the results of BCI competition II.

**5.2.2. Visual Feedback Experiment (ALS Patient).** The performances of DELM on the second BCI competition dataset IB are tested in this section, and this dataset comes from the visual feedback experiment (ALS patient) provided by University of Tuebingen.

The datasets were taken from an artificially respirated ALS patient. The subject was asked to move a cursor up and down on a computer screen, while his cortical potentials were taken. Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor. Each trial lasted 8 s. The visual feedback was presented from second 2 to second 6.5. Only this 4.5-second interval of every trial is provided for training and testing. The sampling rate of 256 Hz and the

TABLE 5: Performance comparison of DELM with the results of BCI competition II on dataset IA.

| #  | Contributor                      | Error (%) |
|----|----------------------------------|-----------|
| 1  | Brett Mensh                      | 11.3      |
| 2  | Guido Dornhege                   | 11.6      |
| 3  | Kai-Min Chung                    | 11.9      |
| 4  | Tzu-Kuo Huang                    | 15.0      |
| 5  | David Pinto                      | 15.7      |
| 6  | Juma Mbwana                      | 17.1      |
| 7  | Vladimir Bostanov                | 17.4      |
| 8  | Ulrich Hoffmann                  | 17.8      |
| 9  | Deniz Erdogmus                   | 19.1      |
| 10 | Justin Sanchez                   | 19.8      |
| *  | Ours (the average value of DELM) | 13.50     |
| *  | Ours (the best value of DELM)    | 8.19      |

recording length of 4.5 s result in 1152 samples per channel for every trial, and the details are presented in Table 6.

As shown in Figure 7, we can make choices that the number of ELM hidden layer nodes on BCI competition II dataset IA is 2000; the regularized parameter  $C$  and the kernel parameter  $\gamma$  of KELM are  $10^{-1}$  and  $10^3$ . The structure of MLELM is 8064-500-500-2000-2, where the parameter  $C$  for layer 8064-500 is  $10^1$ , the parameter  $C$  for layer 500-2000 is  $10^8$ , and the parameter  $C$  for layer 2000-2 is  $10^4$ . The structure of DELM is 8064-500-500-L-2, where the parameter  $C$  for layer 8064-500 is  $10^{-2}$ , the parameter  $C$  for layer L-2 is  $10^{-8}$ , and the kernel parameter  $\gamma$  is  $10^1$ .

The performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IB is shown in Table 7. It is clearly observed that the best of DELM testing accuracy is not lower than MLELM, ELM, and KELM.

TABLE 6: The details of the second BCI competition dataset IB.

| Dataset                       | Number of samples |                 | Number of attributes | Number of labels |
|-------------------------------|-------------------|-----------------|----------------------|------------------|
|                               | Training samples  | Testing samples |                      |                  |
| BCI competition II dataset IB | 200               | 180             | 8064                 | 2                |

TABLE 7: Performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IB.

| Dataset                       | Algorithm | #       | Training accuracy | Testing accuracy | Training time (s) | Testing time (s) |
|-------------------------------|-----------|---------|-------------------|------------------|-------------------|------------------|
| BCI competition II dataset IB | Basic ELM | Average | 1.0000 ± 0        | 0.5172 ± 0.0395  | 3.4636 ± 0.1255   | 2.0602 ± 0.0670  |
|                               |           | Best    | 1.0000            | 0.6056           | —                 | —                |
|                               | KELM      | —       | 1.0000            | 0.5333           | 0.0738            | 0.1285           |
|                               | ML-ELM    | Average | 0.6145 ± 0.0306   | 0.5219 ± 0.0284  | 10.4225 ± 0.1134  | 0.3970 ± 0.0182  |
|                               |           | Best    | 0.6750            | 0.5833           | —                 | —                |
|                               | DELM      | Average | 0.7151 ± 0.0485   | 0.5211 ± 0.0266  | 8.9814 ± 0.2085   | 0.2603 ± 0.0231  |
| Best                          |           | 0.8450  | 0.6056            | —                | —                 |                  |

TABLE 8: Performance comparison of DELM with the results of BCI competition II on dataset IB.

| # | Contributor                      | error  |
|---|----------------------------------|--------|
| 1 | Vladimir Bostanov                | 45.6%  |
| 2 | Tzu-Kuo Huang                    | 46.7%  |
| 2 | Juma Mbwana                      | 46.7%  |
| 4 | Kai-Min Chung                    | 47.8%  |
| 5 | Xichen Sun                       | 48.3%  |
| 6 | Amir Saffari                     | 53.3%  |
| 7 | Fabien Torre                     | 54.4%  |
| 8 | Brett Mensh                      | 56.1%  |
| * | Ours (the average value of DELM) | 47.89% |
| * | Ours (the best value of DELM)    | 39.44% |

MLELM training time is the longest, and the testing time of MLELM and DELM is less than ELM. The performance comparison of DELM with the results of BCI competition II dataset IA is shown in Table 8. It is clear that the average error value of DELM on BCI competition II dataset IA achieves 47.89%, but the min error value has reduced to 39.44%, which is much lower than the results of BCI competition II.

## 6. Conclusions

This paper explores the application of DELM in EEG classification and makes use of two BCI competition datasets to test the performances of DELM. Experimental results show that DELM has the advantage of the least training time and the good efficiency and DELM is an effective BCI classifier. Although DELM has these advantages, there are some places which should be improved, such as the number of all hidden layer nodes, each hidden layer activation function, and each layer parameter  $C$  that are difficult to determine. In this paper, DELM is used to classify preprocessed EEG data and the feature attributes of preprocessed EEG are not extracted, which has certain effects on the experimental results. Future research is to combine the EEG feature extraction methods and DELM, which will be applied to the EEG classification.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61379101), the National Key Basic Research Program of China (No. 2013CB329502), the Natural Science Foundation of Jiangsu Province (No. BK20130209), and the Fundamental Research Funds for the Central Universities (No. 2013XK10).

## References

- [1] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003.
- [2] J.-B. Zhao and Z.-J. Zhang, "Progress in brain-computer interface based on cortical evoked potential," *Space Medicine & Medical Engineering*, vol. 23, no. 1, pp. 74–78, 2010.
- [3] M. Middendorf, G. McMillan, G. Calhoun, and K. S. Jones, "Brain-computer interfaces based on the steady-state visual-evoked response," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 211–214, 2000.
- [4] Z.-Y. Feng, *EEG Applied Research in Personal Identification and Fatigue Detection*, Beijing University of Posts and Telecommunications, 2013.
- [5] N. Ye, Y.-G. Sun, and X. Wang, "Classification of brain-computer interface signals based on common spatial patterns and K-nearest neighbors," *Journal of Northeastern University*, vol. 30, no. 8, pp. 1107–1110, 2009.
- [6] M. Meng and Z.-Z. Luo, "Hand motion classification based on eye-moving assisted EEG," *Pattern Recognition and Artificial Intelligence*, vol. 25, no. 6, pp. 1007–1012, 2012.
- [7] B.-H. Yang, M.-Y. He, L. Liu, and W.-Y. Lu, "EEG classification based on batch incremental SVM in brain computer interfaces," *Journal of Zhejiang University (Engineering Science)*, vol. 47, no. 8, pp. 1431–1436, 2013.

- [8] Q. Yuan, W. Zhou, S. Li, and D. Cai, "Approach of EEG detection based on ELM and approximate entropy," *Chinese Journal of Scientific Instrument*, vol. 33, no. 3, pp. 514–519, 2012.
- [9] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *Algorithmic Learning Theory*, vol. 6925 of *Lecture Notes in Computer Science*, pp. 18–36, Springer, Berlin, Germany, 2011.
- [10] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.
- [11] Y. Bengio and Y. Lecun, "Scaling learning algorithms towards AI," in *Large-Scale Kernel Machines*, vol. 34, pp. 1–41, 2007.
- [12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *American Association for the Advancement of Science: Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [13] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, article 5947, 2009.
- [14] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [15] M. Norouzi, M. Ranjbar, and G. Mori, "Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 2735–2742, IEEE Press, Miami, Fla, USA, 2009.
- [16] L. L. C. Kasun, H.-M. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent System*, vol. 28, no. 6, pp. 31–34, 2013.
- [17] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.
- [18] E. Cambria, G.-B. Huang, L. L. C. Kasun et al., "Extreme learning machines," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.
- [19] W.-Y. Deng, Q.-H. Zheng, L. Chen, and X.-B. Xu, "Research on extreme learning of neural networks," *Chinese Journal of Computers*, vol. 33, no. 2, pp. 279–287, 2010.
- [20] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [21] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: optimally pruned extreme learning ," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.
- [22] F. M. Pouzols and A. Lendasse, "Evolving fuzzy optimally pruned extreme learning machine for regression problems," *Evolving Systems*, vol. 1, no. 1, pp. 43–58, 2010.
- [23] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, vol. 10, no. 3, pp. 262–266, 1989.
- [24] N. Birbaumer, N. Ghanayim, T. Hinterberger et al., "A spelling device for the paralysed," *Nature*, vol. 398, no. 6725, pp. 297–298, 1999.

## Research Article

# The Optimisation for Local Coupled Extreme Learning Machine Using Differential Evolution

**Yanpeng Qu and Ansheng Deng**

*Information Science and Technology College, Dalian Maritime University, Dalian 116026, China*

Correspondence should be addressed to Yanpeng Qu; [yanpengqu@dmlu.edu.cn](mailto:yanpengqu@dmlu.edu.cn)

Received 13 August 2014; Revised 12 November 2014; Accepted 24 November 2014

Academic Editor: Yi Jin

Copyright © 2015 Y. Qu and A. Deng. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Many strategies have been exploited for the task of reinforcing the effectiveness and efficiency of extreme learning machine (ELM), from both methodology and structure perspectives. By activating all the hidden nodes with different degrees, local coupled extreme learning machine (LC-ELM) is capable of decoupling the link architecture between the input layer and the hidden layer in ELM. Such activated degrees are jointly determined by the associated addresses and fuzzy membership functions assigned to the hidden nodes. In order to further refine the weight searching space of LC-ELM, this paper implements an optimisation, entitled evolutionary local coupled extreme learning machine (ELC-ELM). This method makes use of the differential evolutionary (DE) algorithm to optimise the hidden node addresses and the radiuses of the fuzzy membership functions, until the qualified fitness or the maximum iteration step is reached. The efficacy of the presented work is verified through systematic simulated experimentations in both regression and classification applications. Experimental results demonstrate that the proposed technique outperforms three ELM alternatives, namely, the classical ELM, LC-ELM, and OSFuzzyELM, according to a series of reliable performances.

## 1. Introduction

Due to the significant efficiency and simple implementation, extreme learning machine (ELM) [1, 2] has recently enjoyed much attention as a powerful tool in regression and classification applications (e.g., [3, 4]). A variety of the extensions of ELM, therefore, have been developed in an attempt to improve their performances. In general, there are two manners: one is to optimise the methodology of ELM (e.g., online sequential ELM [5] and evolutionary ELM [6]); the other is to refine the hidden layer of ELM for optimising the learning model (e.g., incremental ELM [7], pruned-ELM [8], and two-stage ELM [9]). Several promising performances have been observed through these two schemes, at both theoretical and empirical levels.

Local coupled extreme learning machine (LC-ELM) ultimately develops the classical ELM algorithm by assigning an address to each hidden node in the input space. Given a learning sample, the hidden nodes will be activated at different levels in accordance with the distances from their locations to the input sample. In so doing, the fully coupled

architecture between the input layer and the hidden layer in ELM gets simplified. And the complexity of the weight searching space will be reduced correspondingly. In fact, when the input information is modified, only those highly relevant hidden nodes will be influenced. This process is similar to the learning process of a brain: when a new learning sample is achieved, only relative knowledge needs to be revised with different memory inspired degrees.

In LC-ELM, the addresses and the window radiuses are preset empirically or randomly at present. However, the existence of the nonoptimal addresses and radiuses may yield an inappropriate underlying model, by accident. As a type of metaheuristics, the differential evolution (DE) approach [10] entails few or no assumptions regarding the problem being optimized and has the ability to search for the candidate solutions in very large spaces. In this case, this paper presents an approach termed evolutionary local coupled extreme learning machine (ELC-ELM). The proposed method makes use of DE in an attempt to address the challenges raised by the stochastically predetermined addresses and radiuses. Specifically, in ELC-ELM, DE is utilised to

optimise the addresses and radiuses, according to the resulting root mean squared error (RMSE). Hence, the associated activation degrees are improved. This optimisation procedure is capable of searching for a superior framework of ELC-ELM, until the qualified fitness (consisting of the addresses and radiuses) or the maximum iteration step is reached. To evaluate the performance of this approach, comparative studies between ELC-ELM and the alternative ELM-based techniques (including the classical ELM, LC-ELM, and OSFuzzyELM [11]) are also presented through systematic experimental investigations. The results demonstrate that the proposed work entails improved performances in both regression and classification applications.

The remainder of this paper is structured as follows. An outline of the relevant background materials is presented in Section 2, including LC-ELM and the differential evolution algorithm. The optimisation of LC-ELM, termed evolutionary local coupled extreme learning machine (ELC-ELM), is then described in Section 3. In Section 4, the systematical comparisons between ELC-ELM and several relevant ELM-based algorithms (ELM, LC-ELM, and OSFuzzyELM) are carried out in an experimental evaluation. Section 5 concludes the paper with a short discussion of the potential further works.

## 2. Theoretical Background

For completeness, the basic ideas of local coupled extreme learning machine and differential evolution (DE) [10] are briefly recalled first.

**2.1. Local Coupled Extreme Learning Machine.** Conventionally, extreme learning machine (ELM) algorithms [1, 2] are implemented with a fully coupled framework as, in general, single input activates all hidden nodes. Such structure leads to the computation cost in proportion with the scale of a given network. In LC-ELM, a strategy to decouple the framework linking the input layer to the hidden layer in ELM was proposed. Different from the classical ELM, LC-ELM introduces a parameter, termed ‘‘address,’’ to each hidden node in the input space. Given a learning sample, the distances from the hidden nodes to the input sample are gauged by the fuzzy membership functions as the activated degree of the relevant hidden nodes. Due to the utilisation of these two improvements, this strategy implements the structural simplification of the weight searching space in LC-ELM.

For a dataset which contains  $M$  distinct objects  $(\mathbf{x}_i, \mathbf{t}_i)$ , where  $\mathbf{x}_i \in \mathbb{R}^p$  and  $\mathbf{t}_i \in \mathbb{R}^q$ , the output of an  $N$ -hidden-node nonlinear LC-ELM is

$$\sum_{j=1}^N \beta_j g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) F(S(\mathbf{x}_i, \mathbf{d}_j)), \quad i = 1, \dots, M, \quad (1)$$

where  $g(\cdot)$  denotes the activation function.  $\mathbf{w}_j, b_j$ , and  $\beta_j$  are the network weights.  $F(\cdot)$  is a fuzzy membership function.  $S(\mathbf{x}_i, \mathbf{d}_j)$  is the similarity between the  $i$ th input and the  $j$ th hidden node.  $\mathbf{d}_j \in \mathbb{R}^p$  is the address of the  $j$ th hidden node.

In LC-ELM, the fuzzy membership function  $F(\cdot)$  is defined with the following properties:

- (1)  $F(\cdot)$  is a nonnegative piecewise continuous function,
- (2)  $F(\cdot)$  is monotonically decreasing in  $[0, +\infty)$ ,
- (3)  $F(0) = 1$ ,
- (4)  $F(x) \rightarrow 0, x \rightarrow +\infty$ .

Here,  $F(\cdot)$  is said to be piecewise continuous if it has only a finite number of discontinuities in any interval, and its left and right limits are defined (not necessarily equal) at each discontinuity [2]. In order to adjust the width of the activated area, the underlying radius parameter  $r$  is employed in  $F(\cdot)$ .

Note that, in (1), when the  $F(\cdot)$  is a constant function which is equal to 1, LC-ELM is reduced to the classical ELM. Moreover, when  $\mathbf{w}_j$  in (1) is equal to zero, the fuzzy membership function  $F(\cdot)$  is nonconstant, and the similarity function  $S(\mathbf{x}, \mathbf{d})$  is determined by the norm distance  $\|\mathbf{x} - \mathbf{d}\|$ ; then, the framework of LC-ELM is reduced to the ELM with RBF hidden nodes [2]. In [12], both of these two cases of ELM are proven to own universal approximation capabilities. Therefore, it is reasonable to consider that, for an arbitrary multivariate continuous function, LC-ELM may have the ability to approximate the function under a given accuracy.

For the linear system generated by LC-ELM,

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \quad (2)$$

the hidden-layer output matrix in LC-ELM is

$$\mathbf{H} = [h_{ij} = g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) F(S(\mathbf{x}_i, \mathbf{d}_j))]_{M \times N}, \quad (3)$$

$$i = 1, \dots, M, \quad j = 1, \dots, N.$$

$\boldsymbol{\beta} = [\beta_1, \dots, \beta_N]_{N \times q}^T$  is the matrix of output weights and  $\beta_i$  denotes the weight vector connecting the  $i$ th hidden node and the output layer.  $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_M]_{M \times q}^T$  is the matrix of target outputs. Given such presentation, in the initialisation phase of LC-ELM, the hidden node address  $\mathbf{d}_j$  as well as the hidden layer parameters  $(\mathbf{w}_j, b_j)$  is assigned randomly as well.

Following the above discussion, a three-step LC-ELM algorithm can be summarised in Algorithm 1.

**2.2. Differential Evolution.** Differential evolution (DE) [10] is known as one of the most efficient evolutionary algorithms [13]. It has been widely used to tune the parameters in neural networks [14, 15]. Given a set of parameter vectors  $\{\boldsymbol{\theta}_{k,G} \mid k = 1, 2, \dots, NP\}$  as a population at each generation  $G$ , the basic learning process of DE involves the iteration of the following procedures.

(i) *Mutation.* For each target vector  $\boldsymbol{\theta}_{k,G}$ ,  $k = 1, 2, \dots, NP$ , a mutant vector is generated according to

$$\boldsymbol{\nu}_{k,G+1} = \boldsymbol{\theta}_{r_1,G} + F \cdot (\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G}) \quad (4)$$

with random and mutually different indices  $r_1, r_2, r_3 \in \{1, 2, \dots, NP\}$  and  $F \in [0, 2]$ . The constant factor  $F$  is used to control the amplification of the differential variation  $(\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G})$ .

**Require:**

- $\mathbb{N}$ , the training set  $\{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbb{R}^p, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \dots, M\}$ ,  
 $g$ , the activation function,  
 $S$ , the similarity function,  
 $F$ , the fuzzy membership function,  
 $N$ , the number of hidden nodes,  
 $r$ , the radius of fuzzy membership functions.  
(1) Randomly assign hidden node parameters  $(\mathbf{w}, \mathbf{b})$  and the hidden node address  $\mathbf{d}$ .  
(2) Calculate the hidden layer output matrix  $\mathbf{H}$ .  
(3) Calculate the output weight  $\boldsymbol{\beta}$ .

ALGORITHM 1: Local coupled extreme learning machine.

(ii) *Crossover*. In this procedure, the  $D$ -dimensional trial vector

$$\boldsymbol{\mu}_{k,G+1} = (\boldsymbol{\mu}_{1k,G+1}, \boldsymbol{\mu}_{2k,G+1}, \dots, \boldsymbol{\mu}_{Dk,G+1}) \quad (5)$$

is formed such that

$$\boldsymbol{\mu}_{k,G+1} = \begin{cases} \boldsymbol{\nu}_{lk,G+1} & \text{if } \text{rand } b(l) \leq CR \text{ or } l = \text{rnbr}(k) \\ \boldsymbol{\theta}_{lk,G} & \text{if } \text{rand } b(l) \leq CR \text{ or } l \neq \text{rnbr}(k), \end{cases} \quad (6)$$

where  $\text{rand } b(l)$  is the  $l$ th evaluation of a uniform random number generator with an outcome in  $[0, 1]$ ,  $CR$  is the crossover constant in  $[0, 1]$  which is specified independent of the algorithm, and  $\text{rnbr}(k)$  is a random chosen integer index  $\in$  which ensures that  $\boldsymbol{\nu}_{lk,G+1}$  obtains at least one parameter from  $\boldsymbol{\nu}_{lk,G+1}$ .

(iii) *Selection*. If vector  $\boldsymbol{\mu}_{k,G+1}$  is better than  $\boldsymbol{\theta}_{k,G}$ , then  $\boldsymbol{\theta}_{k,G+1}$  is set to  $\boldsymbol{\mu}_{k,G+1}$ . Otherwise, the existing value  $\boldsymbol{\theta}_{k,G}$  is retained as  $\boldsymbol{\theta}_{k,G+1}$ .

Overall, DE is an approach that optimises a problem through iterative attempt of improving a candidate solution with regard to a given measure of quality (i.e., fitness function). As a type of metaheuristics, such strategy entails few or no assumptions regarding the problem being optimised and has the ability to search in the large spaces (such as the weight searching space of LC-ELM) of candidate solutions [10].

### 3. Evolutionary Local Coupled Extreme Learning Machine

In LC-ELM, the strategy to decouple the linking architecture between the input layer and the hidden layer is guided by the predetermined addresses and the radiuses. Such parameters are preset randomly and empirically. However, the existence of the nonoptimal addresses and radiuses may yield an inappropriate model by accident. In order to make an optimisation for these addresses and the radiuses, an evolutionary local coupled extreme learning machine (ELC-ELM) method is hereby exploited in this paper. Such approach considers the tuples of addresses and radiuses as the solutions of an optimisation problem and searches for them by the use of DE.

In so doing, ELC-ELM can expect a more reliable implementation in a variety of applications.

For a dataset which contains  $M$  distinct objects  $(\mathbf{x}_i, \mathbf{t}_i)$ , where  $\mathbf{x}_i \in \mathbb{R}^p$  and  $\mathbf{t}_i \in \mathbb{R}^q$ , the main procedure of an  $N$ -hidden-node ELC-ELM algorithm consists of the following.

(i) *Random Generation of a Population of Individuals*. Each individual in the population is composed of a set of the addresses and radiuses

$$\boldsymbol{\theta} = \{\mathbf{d}, \mathbf{r}\}, \quad (7)$$

where  $\mathbf{d} = \{\mathbf{d}_i \mid \mathbf{d}_i \in \mathbb{R}^p, i = 1, \dots, N\}$  and  $\mathbf{r} \in \mathbb{R}^N$  are initialised within the range of  $[0, 1]$  at random. Then, these parameters are employed to measure the activated degrees of the hidden nodes.

Note that, in this step, the input weights  $\mathbf{w}$  and hidden node biases  $\mathbf{b}$  are chosen within the range of  $[0, 1]$  randomly as well. However, they are excluded in the underlying populations in ELC-ELM.

(ii) *Analytical Computation of the Output Weights for Each Individual*. This step is implemented by the use of the Moore-Penrose generalised inverse as with many other ELM algorithms, instead of running any iterative tuning.

(iii) *Evaluation of Each Individual*. The resulting root mean squared error (RMSE) of ELC-ELM is employed to assess the fitness of the individuals in this method, leading to a fitness value for each individual in the population. The mapping between the datasets and the fitness values is termed as the fitness function below. Specifically, in this paper, the RMSE is defined as

$$E = \sqrt{\frac{\sum_{i=1}^M \left\| \sum_{j=1}^N \boldsymbol{\beta}_j h_{ij} - \mathbf{t}_i \right\|_2^2}{M \times q}}. \quad (8)$$

Here, the parameters are defined the same as those in (2).

(iv) *Application of the Three Steps of DE: Mutation, Crossover, and Selection*. In addition to the RMSE, the norm of the output weights  $\|\boldsymbol{\beta}\|$  is also used as a criterion to be added to reinforce the selection procedure. In so doing, when the differences of the fitness between distinct individuals are

**Require:**

- $\mathbb{N}$ , the training set  $\{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbb{R}^p, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \dots, M\}$ ;  
 $g$ , the activation function;  
 $S$ , the similarity function;  
 $F$ , the fuzzy membership function;  
 $N$ , the number of hidden nodes;  
 $Itermax$ , the preset maximum learning epoch of DE.
- (1) Randomly designate hidden node parameters  $(\mathbf{w}, \mathbf{b})$ , hidden node address  $\mathbf{d}$ , and radius  $r$ .
  - (2)  $Iter = 1$ .
  - (3) **while**  $Iter \leq Itermax$  **do**
  - (4) (1) Calculate the hidden layer output matrix  $\mathbf{H}$ ,  
 (2) Calculate the output weight  $\boldsymbol{\beta}$ ,  
 (3) Adjust  $(\mathbf{d}, r)$  using DE,  
 (4)  $Iter = Iter + 1$ .
  - (5) **end while**
  - (6) (1) Calculate the hidden layer output matrix  $\mathbf{H}$ .  
 (2) Calculate the output weight  $\boldsymbol{\beta}$ .

ALGORITHM 2: Evolutionary local coupled extreme learning machine.

insignificant, the one that leads to the minimum  $\|\boldsymbol{\beta}\|$  is selected.

(v) *Determination of a New Population  $\boldsymbol{\theta}_{i,G+1}$ .* This is computed as follows:

$$\boldsymbol{\theta}_{k,G+1} = \begin{cases} \boldsymbol{\mu}_{k,G} & \text{if } f(\boldsymbol{\theta}_{k,G}) - f(\boldsymbol{\mu}_{k,G}) > \varepsilon f(\boldsymbol{\theta}_{k,G}), \\ \boldsymbol{\mu}_{k,G} & \text{if } |f(\boldsymbol{\theta}_{k,G}) - f(\boldsymbol{\mu}_{k,G})| < \varepsilon f(\boldsymbol{\theta}_{k,G}), \\ \boldsymbol{\beta}^{\boldsymbol{\mu}_{k,G}} & \text{if } \|\boldsymbol{\beta}^{\boldsymbol{\mu}_{k,G}}\| < \|\boldsymbol{\beta}^{\boldsymbol{\theta}_{k,G}}\|, \\ \boldsymbol{\theta}_{k,G} & \text{else,} \end{cases} \quad (9)$$

where  $f(\cdot)$  is the fitness function (RMSE) and  $\varepsilon$  is the tolerance rate.

(vi) *Iteration of the Above DE Process Once the New Population Is Generated, until the Goal Is Met or the Predetermined Maximum Number of Learning Iterations Is Reached.* Following the above discussion, the ELC-ELM algorithm is summarised in Algorithm 2.

The same as LC-ELM, the implementation of ELC-ELM is highly flexible in dealing with a variety of problems. Specifically, a collection of certain commonly used similarity measures [16, 17] are listed as follows:

- (i)  $p$ -norms:  $S(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p$ , ( $p = 1, 2, +\infty$ ),
- (ii) fuzzy similarity:  $S(\mathbf{x}, \mathbf{y}) = T_{a \in P} \{\mu_{R_a}(\mathbf{x}, \mathbf{y})\}$ ,  
 where  $T$  is a  $T$ -norm,  $P$  is a subset of features, and  $\mu_{R_a}(\mathbf{x}, \mathbf{y})$  is the degree to which objects  $\mathbf{x}$  and  $\mathbf{y}$  are similar for feature  $a$ ,
- (iii) kernel functions:
  - (a) Gaussian kernel:  $S(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\theta)$ ,
  - (b) wave kernel:  $S(\mathbf{x}, \mathbf{y}) = (\theta/\|\mathbf{x} - \mathbf{y}\|) \sin(\|\mathbf{x} - \mathbf{y}\|/\theta)$ ,
  - (c) polynomial kernel:  $S(\mathbf{x}, \mathbf{y}) = (a\mathbf{x} \cdot \mathbf{y} + c)^d$ .

As well as the similarity relations, the fuzzy membership functions in ELC-ELM also enjoy a variety of implementations. For instance, Gaussian function equation (10), the reversed sigmoid function equation (11), and reversed tanh function equation (12) are alternatives in practice:

$$F(x) = \exp\left(-\frac{x^2}{r}\right), \quad (10)$$

$$F(x) = \frac{2}{1 + \exp(x/r)}, \quad (11)$$

$$F(x) = \tanh\left(-\frac{x}{r}\right) + 1. \quad (12)$$

## 4. Experimental Evaluation

This section presents a systematic evaluation of ELC-ELM experimentally. The results and discussions are divided into three parts. Each of them is carried out as a comparison between ELC-ELM and three alternative ELM algorithms: the classical ELM, OSFuzzyELM [11], and LC-ELM. OSFuzzyELM is a variance of ELM which is based on the fuzzy rules.

The first part evaluates ELC-ELM in the aspect of function approximation. The comparison on real-world regression problems is performed in the second part. The third part provides an investigation of the classification performance of ELC-ELM on several benchmark datasets. Note that the fuzzy membership functions and the similarity relations in the following OSFuzzyELM, LC-ELM, and ELC-ELM methods are assigned empirically.

4.1. *Function Regression.* This task is to approximate the Gabor function

$$G(x, y) = \frac{1}{2\pi \times 0.5^2} \exp\left(-\frac{x^2 + y^2}{2 \times 0.5^2}\right) \cos(2\pi(x + y)). \quad (13)$$

TABLE 1: Configurations of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM.

| Configuration                           | ELM             | OSFuzzyELM | LC-ELM                   | ELC-ELM         |
|-----------------------------------------|-----------------|------------|--------------------------|-----------------|
| Input weights<br>&& hidden layer biases | RN in $[-1, 1]$ | N/A        | RN in $[-1, 1]$          | RN in $[-1, 1]$ |
| Activation function                     | Sigmoid         | N/A        | Sigmoid                  | Sigmoid         |
| Hidden node address<br>&& window radius | N/A             | N/A        | RN in $[0, 1]$<br>&& 0.4 | RN in $[0, 1]$  |
| Similarity                              | N/A             | N/A        | Wave kernel              | Wave kernel     |
| Fuzzy membership function               | N/A             | (10)       | (11)                     | (11)            |

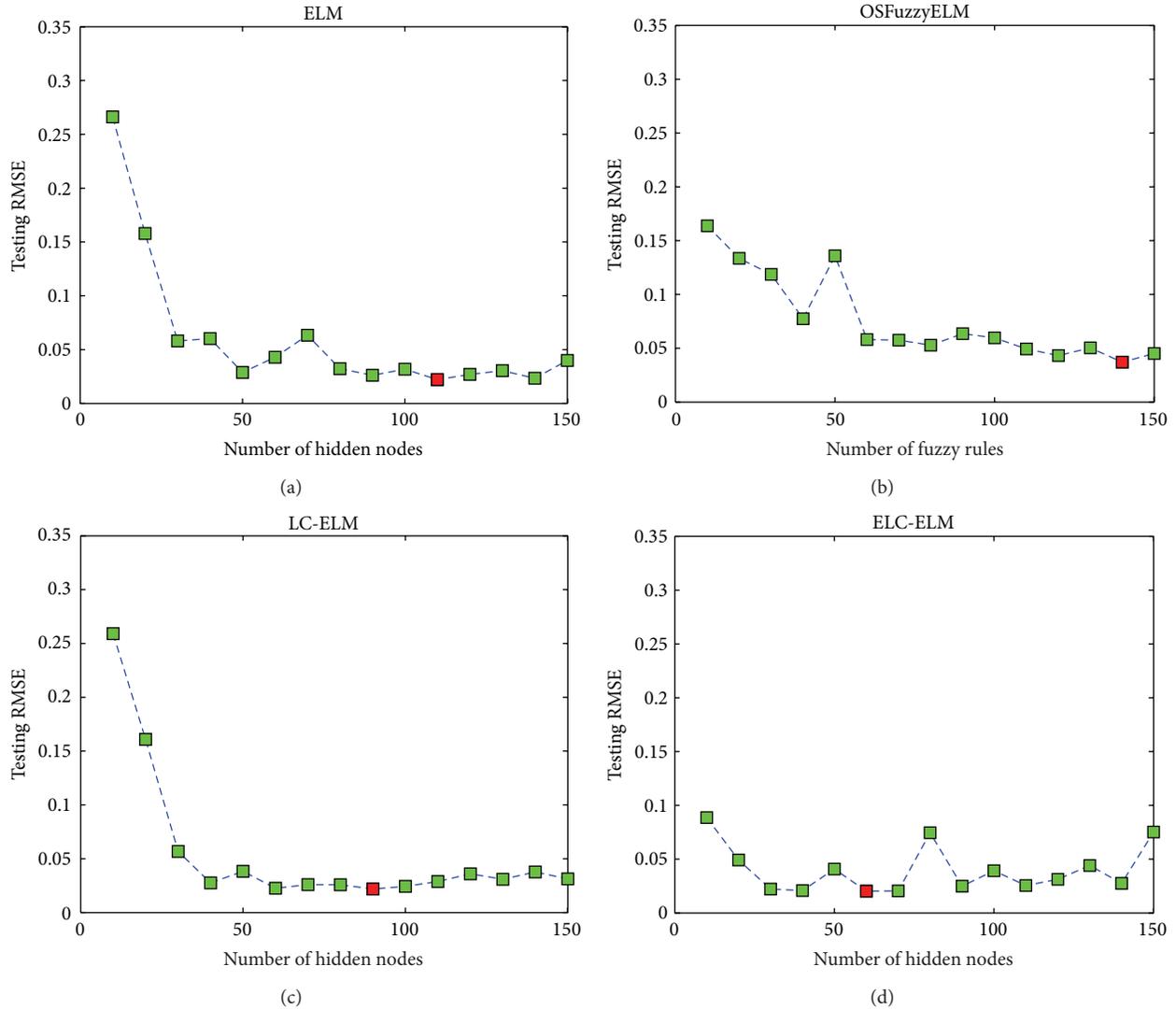


FIGURE 1: Testing RMSEs of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM with respect to different numbers of hidden nodes.

In this example,  $51 \times 51$  training and testing patterns are stochastically selected from a  $[-0.5, 0.5] \times [-0.5, 0.5]$  square region, respectively. The specific configurations of the algorithms involved in this experiment are introduced in Table 1. For simplicity, RN stands for “random number.”

A series of experiments are carried out in order to ascertain the variation in the resulting regression RMSEs by

changing the number of the hidden nodes, or fuzzy rules, in the relevant algorithms. It is noteworthy that the number of hidden nodes (or fuzzy rules) is the tens ranging from 10 to 150. For each of these values, 10 trials have been conducted for the four ELM approaches. The average testing RMSEs are illustrated in Figure 1. Furthermore, across all the 15 numbers of the hidden nodes of each method, the lowest result is

TABLE 2: Function approximation results of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM.

| Algorithm  | Lowest testing | Hidden nodes | Avg. training |        | Avg. testing |        |
|------------|----------------|--------------|---------------|--------|--------------|--------|
|            | RMSE           |              | RMSE          | SD     | RMSE         | SD     |
| ELM        | 0.0221         | 110          | 0.1303        | 0.0054 | 0.0607       | 0.0217 |
| OSFuzzyELM | 0.0371         | 140          | 0.1302        | 0.0068 | 0.0764       | 0.0289 |
| LC-ELM     | 0.0219         | 90           | 0.1295        | 0.0055 | 0.0552       | 0.0173 |
| ELC-ELM    | 0.0203         | 60           | 0.1167        | 0.0015 | 0.0403       | 0.0277 |

denoted by the red marker. As well as such results, the associated numbers of hidden nodes, the average training and testing results, and the standard deviations (SD) of the four ELM-based algorithms are summarised in Table 2.

Overall, it can be observed from Figure 1 and Table 2 that ELC-ELM consistently outperforms the alternative methods with less hidden nodes at the levels of both the lowest and the average RMSEs. In particular, even with the 10 hidden nodes/rules, ELC-ELM is still able to result in a comparable performance (RMSE = 0.0887).

**4.2. Real-World Regression Problems.** This section presents the comparative studies between the proposed approach and the other ELM algorithms on the benchmark regression datasets taken from UCI Machine Learning Repository [18] and Statlib [19]. The specifications of these datasets are shown in Table 3.

In this experiment, 10 trials are conducted for each problem. The training and testing data of the corresponding datasets are reshuffled at each trial of simulation. The configurations of the testing ELM algorithms are roughly the same as those in Table 1. However, the fuzzy membership functions of LC-ELM and ELC-ELM are reversed tanh function (12), and the classical ELM algorithm is constructed with RBF hidden nodes with multiquadric function  $g(x) = (\|x - a\|^2 + b^2)^{1/2}$ . The average RMSE, the corresponding standard deviation (SD), the average training/testing time, and the number of hidden nodes or fuzzy rules, over the training data and the testing data across the 10 trials, are listed in Table 4.

In Table 4, the superior RMSE results, which are lower than their counterparts by more than 0.005, will be shown in boldface. It can be seen from these results that, compared to the remaining three ELM algorithms, ELC-ELM performs better with the lowest RMSE and SD results in general. In particular, ELC-ELM gains significant improvements compared to the others for all datasets except the *Abalone* dataset. Occasionally, for *Autoprice* and *CPU* datasets, the training RMSE results of OSFuzzyELM are better than those of ELC-ELM. However, given the associated testing RMSE results, this significance may be caused by the overfitting that happened to OSFuzzyELM. Although, due to the complexity of DE, the evolution procedure in ELC-ELM is more time consuming than the conventional ones, the generalisation ability of ELC-ELM is improved.

**4.3. Classification Problems.** In this section, the classification performances of ELC-ELM will be compared against those

TABLE 3: Specifications of tested regression problems.

| Dataset   | Number of attributes | Number of training data | Number of testing data |
|-----------|----------------------|-------------------------|------------------------|
| Abalone   | 8                    | 2784                    | 1393                   |
| Autoprice | 9                    | 106                     | 53                     |
| Bodyfat   | 14                   | 168                     | 84                     |
| Computer  | 12                   | 5461                    | 2731                   |
| CPU       | 6                    | 139                     | 70                     |
| Housing   | 13                   | 377                     | 169                    |

of ELM, OSFuzzyELM, and LC-ELM on several benchmark datasets [18]. The specifications of the datasets are displayed in Table 5.

Again, in this experiment, each problem will run 10 trails with reshuffling the training and testing data. Different from the configuration in Section 4.2, the sigmoid hidden nodes will be adopted in ELM and the window radius in LC-ELM is fixed to be 0.7 empirically. For these four ELM-based methods, the average classification accuracy (Accy), the standard deviation (SD), the average training/testing time, and the numbers of hidden nodes or fuzzy rules, over the training data and the testing data across the 10 trials, are listed in Table 6. The same numbers of hidden nodes are used in ELM, LC-ELM, and ELC-ELM.

Likewise, in Table 6, the superior classification accuracies which are higher than their counterparts by more than 0.5% will be marked in boldface. Overall, the LC-ELM method outperforms the other three ELM-based algorithms in all testing results. In particular, for *Ecoli* dataset, ELC-ELM also results in the best training accuracy. This indicates that the model of ELC-ELM enjoys a remarkable generalisation. Although OSFuzzyELM yields the better training results performance on 3 of 6 datasets, given the corresponding testing results, it suffers from overfitting. Again, the ELC-ELM costs more time to implement the classification models which perform better for testing data.

In summary, examining all of the results obtained, it is clear that, due to an evolved weight searching space by DE, ELC-ELM is more reliable than the others in addressing the regression and classification problems. Although, since DE is adopted in ELC-ELM, the evolution procedure is more time consuming than the conventional ones, the resulting model enjoys a greater generalisation ability. Even with a small number of hidden nodes, ELC-ELM still has the ability to gain certain considerable performances. Additionally, given

TABLE 4: Regression results of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM.

| Dataset   | Algorithm  | Training (%)  |        | Training time        | Testing (%)   |        | Testing time | Number of nodes/rules |
|-----------|------------|---------------|--------|----------------------|---------------|--------|--------------|-----------------------|
|           |            | RMSE          | SD     |                      | RMSE          | SD     |              |                       |
| Abalone   | ELM        | 0.0761        | 0.0012 | 0.0172               | 0.0776        | 0.0027 | 0.0016       | 25                    |
|           | OSFuzzyELM | 0.0741        | 0.0015 | 1.3026               | 0.0766        | 0.0023 | 0.0406       | 5                     |
|           | LC-ELM     | 0.0754        | 0.0011 | 0.4524               | 0.0767        | 0.0018 | 0.2699       | 25                    |
|           | ELC-ELM    | 0.0754        | 0.0013 | 537.1114             | 0.0743        | 0.0025 | 0.2278       | 25                    |
| Autoprice | ELM        | 0.0879        | 0.0101 | 0.0016               | 0.1205        | 0.0190 | 0.0009       | 15                    |
|           | OSFuzzyELM | <b>0.0401</b> | 0.0052 | 0.0624               | 0.1096        | 0.0078 | 0.0252       | 3                     |
|           | LC-ELM     | 0.0757        | 0.0044 | 0.0031               | 0.0842        | 0.0087 | 0.0016       | 15                    |
|           | ELC-ELM    | 0.0805        | 0.0059 | 37.1688              | <b>0.0769</b> | 0.0048 | 0.0025       | 15                    |
| Bodyfat   | ELM        | 0.0678        | 0.0038 | 0.0062               | 0.1295        | 0.0191 | 0.0047       | 50                    |
|           | OSFuzzyELM | 0.0790        | 0.0038 | 0.0686               | 0.1264        | 0.0323 | 0.0031       | 2                     |
|           | LC-ELM     | 0.0661        | 0.0038 | 0.0562               | 0.1243        | 0.0243 | 0.0218       | 50                    |
|           | ELC-ELM    | 0.0680        | 0.0013 | 193.9529             | <b>0.1129</b> | 0.0115 | 0.0224       | 50                    |
| Computer  | ELM        | 0.0339        | 0.0008 | 0.3354               | 0.0408        | 0.0044 | 0.0406       | 125                   |
|           | OSFuzzyELM | 0.0257        | 0.0006 | 83.6555              | 0.0346        | 0.0044 | 0.1950       | 15                    |
|           | LC-ELM     | 0.0345        | 0.0016 | 2.2511               | 0.0407        | 0.0033 | 0.9812       | 125                   |
|           | ELC-ELM    | 0.0279        | 0.0003 | $5.6467 \times 10^3$ | <b>0.0288</b> | 0.0005 | 1.0868       | 125                   |
| CPU       | ELM        | 0.0476        | 0.0066 | 0.0016               | 0.0865        | 0.0584 | 0.0008       | 10                    |
|           | OSFuzzyELM | <b>0.0284</b> | 0.0035 | 0.0499               | 0.0659        | 0.0339 | 0.0031       | 3                     |
|           | LC-ELM     | 0.0416        | 0.0070 | 0.0047               | 0.0582        | 0.0203 | 0.0016       | 10                    |
|           | ELC-ELM    | 0.0394        | 0.0069 | 29.3719              | <b>0.0360</b> | 0.0092 | 0.0025       | 10                    |
| Housing   | ELM        | 0.0793        | 0.0047 | 0.0062               | 0.0929        | 0.0093 | 0.0016       | 50                    |
|           | OSFuzzyELM | 0.0645        | 0.0043 | 0.3011               | 0.0924        | 0.0163 | 0.0094       | 5                     |
|           | LC-ELM     | 0.0711        | 0.0035 | 0.1076               | 0.0884        | 0.0091 | 0.0390       | 50                    |
|           | ELC-ELM    | 0.0682        | 0.0040 | 281.3338             | <b>0.0807</b> | 0.0127 | 0.3768       | 50                    |

TABLE 5: Specifications of tested classification problems.

| Dataset      | Number of attributes | Number of training data | Number of testing data | Number of classes |
|--------------|----------------------|-------------------------|------------------------|-------------------|
| <i>Ecoli</i> | 7                    | 224                     | 112                    | 8                 |
| Glass        | 9                    | 142                     | 72                     | 7                 |
| Ionosphere   | 34                   | 153                     | 77                     | 2                 |
| Iris         | 4                    | 100                     | 50                     | 3                 |
| Sonar        | 60                   | 138                     | 70                     | 2                 |
| Wisconsin    | 9                    | 455                     | 228                    | 2                 |

the large number of the similarity relations and the fuzzy membership functions, ELC-ELM can be implemented into various forms, the same as LC-ELM. This mechanism allows ELC-ELM to have the ability to generate solutions for different problems flexibly.

## 5. Conclusion

This paper has presented an approach entitled evolutionary local coupled extreme learning machine (ELC-ELM), in an attempt to address the challenges raised by the stochastically predetermined addresses and radiuses of LC-ELM. The existence of such nonoptimal parameters may yield an inappropriate model of LC-ELM, accidentally. In ELC-ELM, the differential evolution (DE) algorithm is utilised to optimise this tuple (address and radius) and the associated activated

degrees, according to the resulting root mean squared errors. This optimisation procedure will improve the underlying model of ELC-ELM, until the satisfactory solution (population) or the maximum iteration step is reached. Due to the massive existence of the fuzzy membership functions and the similarity relations, the implementation of ELC-ELM is highly flexible. Experimental results demonstrate that the proposed algorithm entails better performances, compared to three alternative ELM-based approaches.

Though promising, further research will help strengthen the potential of the proposed approach. In particular, due to the use of DE, as the scale of the problem increases, the training progress of ELC-ELM will become more time consuming than the alternative methods in this paper. Although ELC-ELM enjoys a significant generalisation ability, the efficiency of ELC-ELM still requires enhancing in the future. Topics for

TABLE 6: Classification results of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM.

| Dataset    | Algorithm  | Training (%) |      | Training time | Testing (%)  |      | Testing time | Number of nodes/rules |
|------------|------------|--------------|------|---------------|--------------|------|--------------|-----------------------|
|            |            | Accy         | SD   |               | Accy         | SD   |              |                       |
| Ecoli      | ELM        | 89.78        | 0.90 | 0.0062        | 85.71        | 1.68 | 0.0031       | 20                    |
|            | OSFuzzyELM | 90.04        | 1.53 | 0.1045        | 86.79        | 2.72 | 0.0047       | 5                     |
|            | LC-ELM     | 89.29        | 1.17 | 0.0218        | 87.14        | 4.23 | 0.0156       | 20                    |
|            | ELC-ELM    | <b>90.89</b> | 0.74 | 312.5917      | <b>89.29</b> | 4.29 | 0.0162       | 20                    |
| Glass      | ELM        | 72.25        | 3.82 | 0.0047        | 63.47        | 5.36 | 0.0016       | 20                    |
|            | OSFuzzyELM | <b>92.04</b> | 1.88 | 0.1529        | 63.33        | 4.05 | 0.0062       | 10                    |
|            | LC-ELM     | 74.79        | 2.97 | 0.0172        | 63.75        | 6.56 | 0.0062       | 20                    |
|            | ELC-ELM    | 73.59        | 3.31 | 21.9727       | <b>66.39</b> | 6.98 | 0.0074       | 20                    |
| Ionosphere | ELM        | 91.70        | 1.69 | 0.0078        | 83.51        | 3.12 | 0.0031       | 40                    |
|            | OSFuzzyELM | <b>95.88</b> | 3.29 | 0.1357        | 80.91        | 4.98 | 0.0031       | 3                     |
|            | LC-ELM     | 93.86        | 2.05 | 0.0359        | 86.49        | 2.68 | 0.0218       | 40                    |
|            | ELC-ELM    | 93.53        | 1.70 | 121.4764      | 86.84        | 3.51 | 0.0236       | 40                    |
| Iris       | ELM        | 98.30        | 0.67 | 0.0031        | 96.60        | 2.50 | 0.0016       | 15                    |
|            | OSFuzzyELM | 98.50        | 0.71 | 0.0296        | 96.00        | 2.11 | 0.0031       | 5                     |
|            | LC-ELM     | 97.90        | 1.29 | 0.0078        | 96.40        | 2.07 | 0.0047       | 15                    |
|            | ELC-ELM    | 97.40        | 0.52 | 11.8374       | <b>97.20</b> | 2.70 | 0.0054       | 15                    |
| Sonar      | ELM        | 88.70        | 2.92 | 0.0031        | 75.57        | 3.95 | 0.0016       | 40                    |
|            | OSFuzzyELM | <b>97.46</b> | 3.35 | 0.2558        | 67.00        | 8.21 | 0.0062       | 3                     |
|            | LC-ELM     | 89.42        | 4.06 | 0.0343        | 76.29        | 6.25 | 0.0187       | 40                    |
|            | ELC-ELM    | 89.78        | 2.95 | 117.0569      | <b>78.00</b> | 6.61 | 0.0178       | 40                    |
| Wisconsin  | ELM        | 97.71        | 0.55 | 0.0047        | 96.36        | 1.42 | 0.0031       | 40                    |
|            | OSFuzzyELM | 97.54        | 0.38 | 0.1934        | 96.45        | 1.10 | 0.0094       | 5                     |
|            | LC-ELM     | 97.49        | 0.43 | 0.0874        | 96.97        | 1.06 | 0.0406       | 40                    |
|            | ELC-ELM    | 97.41        | 0.53 | 250.2381      | <b>97.54</b> | 1.59 | 0.0397       | 40                    |

further research also include a more comprehensive study of how ELC-ELM would perform with other fuzzy membership functions and similarity relations [20] as the alternative. Correspondingly, the sensitivity of these chosen functions is also necessary to be exploited in theory. Furthermore, a more complete comparison of ELC-ELM against the other state-of-the-art learning techniques over different datasets from real application domains [3, 21] would form the basis for a wider series of topics for future studies.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgments

This work is jointly supported by the National Natural Science Foundation of China (61272171), the Fundamental Research Funds for the Central Universities (nos. 3132014094, 3132013325, and 3132013335), and the China Postdoctoral Science Foundation (2013M541213).

### References

- [1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.
- [2] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107-122, 2011.
- [3] Y. Qu, C. Shang, W. Wu, and Q. Shen, "Evolutionary fuzzy extreme learning machine for mammographic risk analysis," *International Journal of Fuzzy Systems*, vol. 13, no. 4, pp. 282-291, 2011.
- [4] Y. Yang, Y. Wang, and X. Yuan, "Bidirectional extreme learning machine for regression problem and its learning effectiveness," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1498-1505, 2012.
- [5] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.
- [6] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.
- [7] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16-18, pp. 3056-3062, 2007.
- [8] H. Rong, Y.-S. Ong, A.-H. Tan, and Z. Zhu, "A fast pruned extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1-3, pp. 359-366, 2008.
- [9] Y. Lan, Y. C. Soh, and G.-B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16-18, pp. 3028-3038, 2010.

- [10] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [11] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "On-line sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.
- [12] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [13] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Oxford, UK, 1996.
- [14] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.
- [15] B. Subudhi and D. Jena, "A differential evolution based neural network approach to nonlinear system identification," *Applied Soft Computing*, vol. 11, no. 1, pp. 861–871, 2011.
- [16] M. G. Genton, "Classes of kernels for machine learning: a statistics perspective," *The Journal of Machine Learning Research*, vol. 2, pp. 299–312, 2001.
- [17] R. Jensen and Q. Shen, "New approaches to fuzzy-rough feature selection," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 824–838, 2009.
- [18] K. Bache and M. Lichman, "UCI machine learning repository," 2013.
- [19] M. Mike, *Statistical Datasets*, Department of Statistics, Carnegie Mellon University, 1989.
- [20] Y. Qu, C. Shang, Q. Shen, N. M. Parthaláin, and W. Wu, "Kernel-based fuzzy-rough nearest neighbour classification," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ '11)*, pp. 1523–1529, June 2011.
- [21] C. Shang, D. Barnes, and Q. Shen, "Facilitating efficient mars terrain image classification with fuzzy-rough feature selection," *International Journal of Hybrid Intelligent Systems*, vol. 8, no. 1, pp. 3–13, 2011.

## Research Article

# Detecting Copy Directions among Programs Using Extreme Learning Machines

**Bin Wang, Xiaochun Yang, and Guoren Wang**

*College of Information Science and Engineering, Northeastern University, Liaoning 110819, China*

Correspondence should be addressed to Xiaochun Yang; [yangxc@mail.neu.edu.cn](mailto:yangxc@mail.neu.edu.cn)

Received 21 August 2014; Revised 10 November 2014; Accepted 10 November 2014

Academic Editor: Tao Chen

Copyright © 2015 Bin Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Because of the complexity of software development, some software developers may plagiarize source code from other projects or open source software in order to shorten development cycle. Many methods have been proposed to detect plagiarism among programs based on the program dependence graph, a graph representation of a program. However, to our best knowledge, existing works only detect similarity between programs without detecting copy direction among them. By employing extreme learning machine (ELM), we construct feature space for describing features of every two programs with possible plagiarism relationship. Such feature space could be large and time consuming, so we propose approaches to construct a small feature space by pruning isolated control statements and removable statements from each program to accelerate both training and classification time. We also analyze the features of data dependencies between any original program and its copy program, and based on it we propose a feedback framework to find a good feature space that can achieve both accuracy and efficiency. We conducted a thorough experimental study of this technique on real C programs collected from the Internet. The experimental results show the high accuracy and efficiency of our ELM-based approaches.

## 1. Introduction

The Internet and open source software are developing rapidly nowadays, providing developers easier accesses to get various open source software code. Meanwhile with the increase of users' need, software developers have to develop more complicated software product, leading to longer development cycle which directly determines development cost as well as enterprise profit. To save development cost, some illegal developers inevitably utilize all possible methods to shorten development cycle. One of the fastest methods is to develop their project on existing projects, open source software, or prototype products. Such behavior can be infringement to original authors. In order to prevent such infringement, a copy detection tool is needed to detect software source code plagiarism.

Many methods have been proposed to detect plagiarism among programs based on the program dependence graph (PDG for short) [1], a graph representation of a program. However, they only focus on detecting similarity but copy

direction between any two similar programs (or PDGs). To our best knowledge, this is the first work to detect both plagiarism and copy direction among programs.

In this paper, we propose a framework based on ELM [2] to detect copy directions among programs. We first classify programs into a set of programs with possible plagiaristic relationship by adopting ELMs. Based on it, we detect copy direction by considering dependencies in programs.

The challenges and contributions of detecting copy directions are as follows.

(i) A source program and its plagiaristic program could be only partially similar since the plagiaristic program might contain some useless statements to make it different from the source program. As we know, finding common statements between two programs or common subgraphs between their corresponding PDGs is very time consuming. In this paper, we propose an extreme learning machine (ELM) based framework to learn this potential similarity and classify PDGs accordingly since ELM is well-known and very efficient for classification with high accuracy.

```

(1) int count(Istream *input, input c, int f1, int f2)
(2) {
(3)     int count, revise, valLeft, valRight, temp, findVal, count;
(4)     valLeft = getCrest(f1);
(5)     valRight = getCrest(f2);
(6)     revise = GlobalVal;
(7)     tmp = c * (valLeft - revise) + valRight;
(8)     findVal = temp/2;
(9)     count = 0;
(10)    char * inputStr;
(11)    while(inputStr = getNext(input))
(12)    {
(13)        int inputVal = paserInt(inputStr);
(14)        if(inputVal == 0)
(15)        {
(16)            int any = findVal + inputVal;
(17)            int any2 = any + findVal;
(18)        }
(19)        if(inputVal == findVal)
(20)            count++;
(21)    }
(22)    return count;
(23) }

```

FIGURE 1: A program.

(ii) A plagiaristic program might contain useless control statements like loops, selective structures, which makes PDGs of a source program and its plagiarism dissimilar and results in low detection accuracy. In order to enhance the accuracy of classification, we further utilize control dependencies in PDGs to remove subgraphs corresponding to those useless control statements and shrink the size of PDGs in Section 4. We show using ELM to classify PDGs with small sizes not only increases the classification accuracy but also decreases classification time, since using smaller PDGs as training set means a smaller feature space, which accelerates both training and classification time. This ELM-based approach greatly exceeds the detection ability of existing algorithms.

(iii) Among a set of programs with plagiaristic relationship, how can we determine which one is the source program and which one copies it? To our best knowledge, this is the first work to solve the problem. In Section 5 we propose a scoring scheme by combining both control dependencies and data dependencies in PDGs to detect copy directions.

In Section 6 we present experimental results on real data sets to demonstrate the accuracy and time efficiency of the proposed technique.

## 2. Preliminary and Background

**2.1. Computer Program.** A computer program is a sequence of statements, written to perform a specific task with a computer. In a program, we mainly focus on two kinds of statements, control statement and other statement. A control statement always generates a boolean value, whereas other statements have no such requirement. Two kinds of dependencies are defined as follows.

(i) *Control Dependency.* A statement  $S$  is control-dependent on a control statement  $C$ , if execution of  $S$  depends on the boolean value of  $C$ .

(ii) *Data Dependency.* A statement  $S_1$  is data-dependent on another statement  $S_2$ , if  $S_1$  reads the value of a variable  $v$  that has been changed by its preceding statement  $S_2$ .

For instance, Figure 1 shows an example program, where statement `inputVal = paserInt(inputStr)` (line 13) is control-dependent on statement `inputStr = getNext(input)` (line 11), because the execution of the former depends on whether the boolean value of the latter is 1. Statement `findVal = temp/2` (line 8) is data-dependent on statement `temp = c * (valLeft - revise) + valRight` (line 7), since the former reads the value of variable `temp`, and the latter does an assignment on `temp`.

**2.2. Program Dependence Graph.** A program dependence graph (PDG) [1] is a graph structure consisting of a program unit, such as statements and relations among variables in a C language function. Given a program  $P$ , let  $G(P)$  be the program dependence graph of  $P$ . Each node in a PDG represents a program statement, and edges between nodes represent dependencies among statements.

*Definition 1.* A PDG is a graph  $G(V, E)$  and it satisfies the following.

- (i) The node set  $V$  in  $G(P)$  represents the set of statements in program  $P$ .
- (ii) The edge set  $E$  in  $G(P)$  represents the set of data dependencies and control dependencies between statements in program  $P$ .

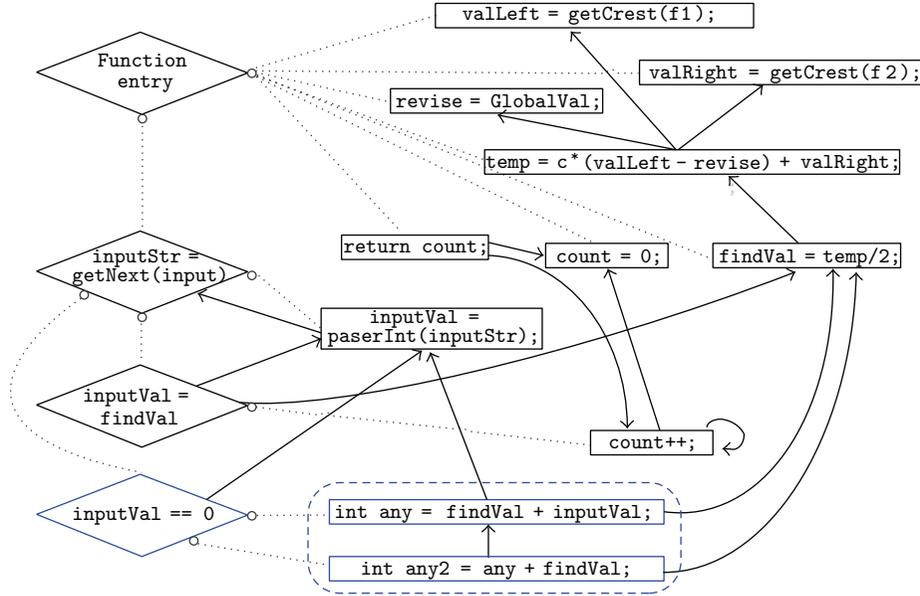


FIGURE 2: The PDG for the program shown in Figure 1.

For example, Figure 2 shows the PDG for the program in Figure 1. The PDG of every function starts from conditional statement “function entry.” Each rhombic node in Figure 2 represents a conditional control statement, and the dotted line with a circle at the node represents a control dependency. Each edge represents a data dependence.

*Problem Description.* Given a set of PDGs  $\{G(P_1), \dots, G(P_n)\}$  corresponding to the set of programs  $\{P_1, \dots, P_n\}$ , find all pairs of PDGs  $\langle G(P_i), G(P_j) \rangle$  such that the program  $P_j$  is a plagiarism of the program  $P_i$ , denoted by  $P_i \rightsquigarrow P_j$  ( $1 \leq i, j \leq n, i \neq j$ ).

**2.3. Extreme Learning Machine (ELM).** ELM [2, 3] has been widely applied in many classification applications [4]. It was originally developed for single hidden layer feedforward neural networks (SLFNs) and then extended to the “generalized” SLFNs where the hidden layer need not be neuron alike [5, 6]. ELM first randomly assigns the input weights and the hidden layer biases and then analytically determines the output weights of SLFNs. It can achieve better generalization performance than other conventional learning algorithms at an extremely fast learning speed. Besides, ELM is less sensitive to user-specified parameters and can be deployed faster and more conveniently [7–9].

For  $N$  arbitrary distinct samples  $(x_j, t_j)$ , where  $\vec{x}_j = [x_{j1}, x_{j2}, \dots, x_{jm}]^T \in \mathbb{R}^n$  and  $\vec{t}_j = [t_{j1}, t_{j2}, \dots, t_{jm}]^T \in \mathbb{R}^m$ , standard SLFNs with  $L$  hidden nodes and activation function  $g(x)$  are mathematically modeled as

$$\sum_{i=1}^L \beta_i g_i(\vec{x}_j) = \sum_{i=1}^L \beta_i g(\vec{w}_i \cdot \vec{x}_j + b_i) = \vec{\delta}_j \quad (1 \leq j \leq N), \quad (1)$$

where  $L$  is the number of hidden layer nodes,  $\vec{w}_i = [w_{i1}, w_{i2}, \dots, w_{im}]^T$  is the weight vector between the  $i$ th hidden node and the input nodes,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the weight vector between the  $i$ th hidden node and the output nodes,  $b_i$  is the threshold of the  $i$ th hidden node, and  $\vec{\delta}_j = [\delta_{j1}, \delta_{j2}, \dots, \delta_{jm}]^T$  is the  $j$ th output vector of the SLFNs [5].

The standard SLFNs with  $L$  hidden nodes and activation function  $g(x)$  can approximate these  $N$  samples with zero error. It means  $\sum_{j=1}^N \|\vec{\delta}_j - \vec{t}_j\| = 0$  and there exist  $\beta_i$ ,  $\vec{w}_i$ , and  $b_i$  such that

$$\sum_{i=1}^L \beta_i g(\vec{w}_i \cdot \vec{x}_j + b_i) = \vec{t}_j \quad (1 \leq j \leq N). \quad (2)$$

Equation (2) can be expressed compactly as follows:

$$\vec{H}\beta = \vec{T}, \quad (3)$$

where  $\vec{H}$  is the hidden layer output matrix of the neural network,  $\vec{H}(\vec{w}_1, \vec{w}_2, \dots, \vec{w}_L, b_1, b_2, \dots, b_L, \vec{x}_1, \vec{x}_2, \dots, \vec{x}_L) =$

$$\begin{bmatrix} h_{11} \\ \vdots \\ h_{Lj} \end{bmatrix} = \begin{bmatrix} g(\vec{w}_1 \cdot \vec{x}_1 + b_1) & g(\vec{w}_2 \cdot \vec{x}_1 + b_2) & \cdots & g(\vec{w}_L \cdot \vec{x}_1 + b_L) \\ g(\vec{w}_1 \cdot \vec{x}_2 + b_1) & g(\vec{w}_2 \cdot \vec{x}_2 + b_2) & \cdots & g(\vec{w}_L \cdot \vec{x}_2 + b_L) \\ \vdots & \vdots & \ddots & \vdots \\ g(\vec{w}_1 \cdot \vec{x}_N + b_1) & g(\vec{w}_2 \cdot \vec{x}_N + b_2) & \cdots & g(\vec{w}_L \cdot \vec{x}_N + b_L) \end{bmatrix}_{N \times L} \quad (4)$$

$\beta = [\beta_1^T, \dots, \beta_L^T]^T_{m \times L}$ , and  $\vec{T} = [\vec{t}_1^T, \dots, \vec{t}_N^T]^T_{m \times N}$ . Algorithm 1 shows the pseudocode of ELM.

**Input:** Training set  $\mathcal{N} = \{(\vec{x}_j, \vec{t}_j) \mid \vec{x}_j \in \mathbb{R}^n (1 \leq j \leq N)\}$ ;  
 Hidden node output function  $g(\vec{w}_i, b_i, \vec{x}_j)$ ;  
 Number of hidden nodes  $L$ ;  
**Output:** Weight vector  $\beta$ ;  
 (1) **for**  $i = 1$  **to**  $L$  **do**  
 (2) Randomly generate hidden node parameters  $(\vec{w}_i, b_i)$ ;  
 (3) Calculate the hidden layer output matrix  $\vec{H}$ ;  
 (4) return the calculated output weight vector  $\beta = \vec{H}^\dagger \vec{T}$ , where  $\vec{H}^\dagger$  is the Moore-Penrose generalized by the inverse of matrix  $H$ ;

ALGORITHM 1: ELM training.

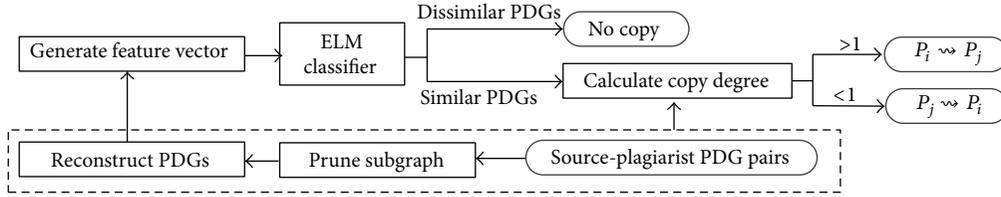


FIGURE 3: An ELM-based framework for determining copy direction.

### 3. An ELM-Based Framework for Determining Copy Direction

To our best knowledge, there is no detection algorithm that can directly compute plagiarism direction correctly. In this section, we propose an ELM-based framework for determining copy directions between any two PDGs.

Figure 3 shows the framework. Generally, a source PDG and its plagiarist PDG should be similar, so given a set of PDGs, we first use an ELM classifier to find all similar PDGs. Then for those similar PDGs, we check copy direction by calculating data dependencies scores as *copy degrees*. According to the calculated copy degree, we determine the copy direction between  $P_i$  and  $P_j$ .

In this section, we first show how to generate feature vectors for the ELM.

For a set of PDGs  $\{G(P_1), \dots, G(P_n)\}$ , using ELM, the task is to learn a prediction rule from the training examples  $\{D_j, y_j\}_{j=1}^n$ , where  $D_j$  is the PDG in the training PDGs and  $y_j \in \{+1, -1\}$  is the associated class label. Any similar PDGs have the same class label. Let  $\mathcal{T}$  be the set of all patterns, that is, the set of all subgraphs included in at least one training PDG. Then each training PDG  $G(P_i)$  is encoded as a  $|\mathcal{T}|$ -dimensional vector  $\vec{x}_j = (x_{j,1}, \dots, x_{j,|\mathcal{T}|})$ ,

$$x_{j,t} = I(t \subseteq G(P_i)), \quad (5)$$

where  $I(\cdot)$  is 1 if the condition inside is true and 0 otherwise. For example, Figure 4(i) shows an example of this feature space. Figures 4(a) and 4(b) show an original code and its plagiaristic code, respectively.  $G(P_1)$  and  $G(P_2)$  are their corresponding PDGs shown in Figures 4(e) and 4(f). For simplicity, we only use a line number in each node to express its corresponding statement. A  $|\mathcal{T}|$ -dimensional vector  $\vec{x}_j$  is shown in Figure 4(i), where every value corresponds to a subgraph pattern in the training set. We then use ELM to train these

training examples  $\{D_j, y_j\}_{j=1}^n$  since ELM is very efficient for large vector space.

Now the question is how to choose patterns to construct  $|\mathcal{T}|$ -dimensional vector as the training set. We choose *frequent subgraph* as patterns, which can be efficiently generated by the existing frequent graph mining algorithm, gIndex [10]. By setting a support threshold  $\rho$ , the algorithm in [10] chooses subgraphs whose supports are larger than or equal to  $\rho$  as frequent patterns. The support threshold is progressively increased when the subgraphs grow large. That is, we use low support for small subgraphs and high support for large subgraphs to avoid the exponential growth of  $|\mathcal{T}|$  (our experimental results are reported in Section 6).

### 4. Achieving High Accuracy with Small Feature Space

Generally choosing large subgraphs as patterns achieves high accuracy of detecting copy relationship, since features of each PDG can be kept. However, it requires a small  $\rho$ , which results in a large number of patterns in  $\mathcal{T}$  for constructing the feature space. Therefore, the detection efficiency is sacrificed as a consequence. Now the challenge is how to shrink feature space to make detection efficient and highly accurate.

We observe that for any two source and plagiarist programs  $P_i \rightsquigarrow P_j$ ,  $P_j$  might contain useless control statements like loops, selective structures, which correspond to useless subgraphs for detection. If  $P_j$  contains a large number of such useless subgraphs,  $G(P_i)$  is dissimilar with  $G(P_j)$  and might not be detected by using ELM classifier.

In this section, we show we can still get a small feature space even when using a large support threshold, since removing such useless control statements only decreases the number of (useless) subgraphs. It is interesting to see that

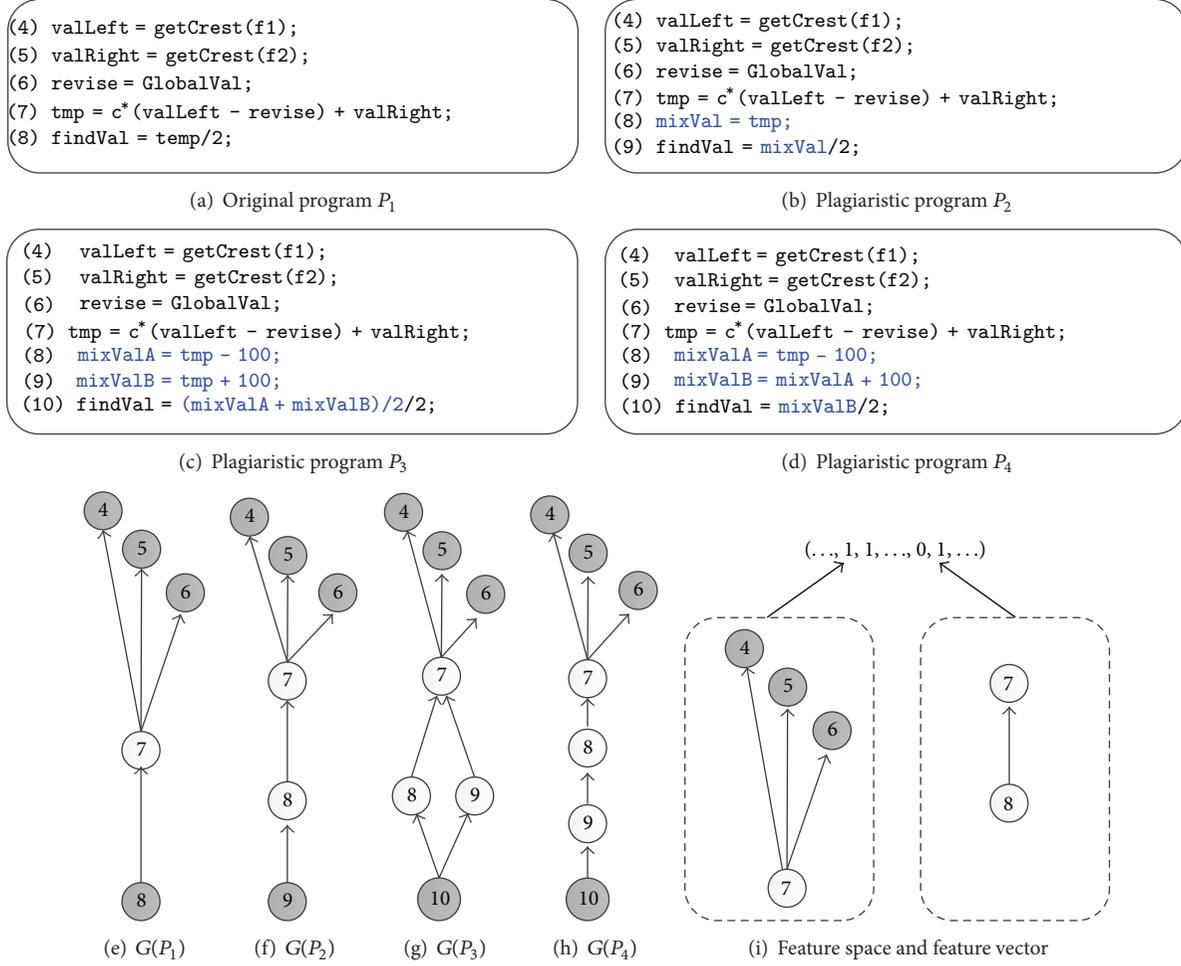


FIGURE 4: An example of changing the order of statements.

decreasing those useless subgraphs will increase the classification accuracy as well as accelerate the training efficiency, since such removal decreases not only noises for detection accuracy but also the number of patterns in  $\mathcal{T}$ .

#### 4.1. Isolated Control Dependence Subgraphs

**Definition 2** (isolated control dependence subgraph). Given a PDG  $G$ , a subgraph  $G'$  of  $G$  is an isolated control dependence subgraph, if the following two conditions hold.

- (1) A variable  $v$  in a statement in  $G'$  is not used by its successive statements.
- (2) For any subgraph  $G''$  of  $G'$ ,  $v$  does not appear in  $G''$ .

For example, in Figure 2, the subgraph marked by dotted line is an isolated control dependence subgraph, whose corresponding statements are useless which could not affect the program's semantics but would change the PDG structure.

For the source-plagiarist PDG pair  $\langle G(P_i), G(P_j) \rangle$ , where  $P_i \rightsquigarrow P_j$ , in order to keep the correctness of execution

semantics of  $P_i$  in  $P_j$ , copyists need to insert or disturb some nodes in  $G(P_i)$  to generate  $G(P_j)$  as follows.

**Feature 1.** A plagiarist PDG  $G(P_j)$  and its source PDG  $G(P_i)$  could be similar locally but dissimilar globally.

For a source PDG  $G(P_i)$  and its copy  $G(P_j)$ ,  $G(P_i)$  and  $G(P_j)$  could share a common subgraph, but globally dissimilar since a copyist may insert some useless control dependence edges in  $G(P_j)$ .

**4.2. Pruning Isolated Control Dependence Subgraphs.** Pruning isolated control dependence subgraphs might affect the execution of original program, because it can affect the function callings. As a consequence, such pruning might also remove subgraphs in an original PDG. Fortunately, according to our observation, this situation has no significant influence on detection result since if a control subgraph in the original program is removed, its corresponding control subgraph in the plagiaristic program would also be removed, and these two programs are still considered to be similar according to graph isomorphism algorithm.

Let  $\lambda$  be the percentage of total removed subgraph nodes in  $V(G)$ . Generally, we are not allowed to remove too many

```

Input: PDG  $G$ , pruning threshold  $\lambda$ ;
Output: PDG after pruning isolated control dependence subgraphs;
(1) Generate control dependence subgraph set  $CG \leftarrow \{g_1, \dots, g_k\}$  in  $G$ ;
(2) foreach control dependence subgraph  $g$  in  $CG$  do
(3)    $g.count \leftarrow 0$ ;
(4)   foreach statement  $S$  in  $g$  do
(5)     if  $S$  has dependence relations with statements outside  $g$  then
(6)        $g.count \leftarrow g.count + 1$ ;
(7) Remove subgraphs in  $CG$  whose number of dependencies are greater than 0;
(8)  $maxNumDelNodes \leftarrow |V(G)| \times \lambda$ ;
(9)  $deletCount \leftarrow 0$ ;
(10)  $G.removeNode \leftarrow 0$ ;
(11) foreach subgraph  $g$  in  $CG$  do
(12)   if  $deletCount + |V(g)| \leq maxNumDelNode$  then
(13)     delete  $g$  from  $G$ ;
(14)      $deletCount \leftarrow deletCount + |V(g)|$ ;
(15)  $G.removeNode \leftarrow deletCount$ ;
(16) return  $G$ ;

```

ALGORITHM 2: Pruning isolated control dependence subgraphs.

isolated control dependence subgraphs so that most nodes can be kept in a PDG. A small  $\lambda$  removes a small number of isolated control dependence subgraphs but leaves large number of useless control statements, which results in low detection recall. Whereas a larger  $\lambda$  could help to detect more plagiarisms but may remove some useless control statements in an original code by mistake. Removing such subgraphs makes detection fast but sacrifices precision (we report the effects of different  $\lambda$  values on real programs in Section 6).

Algorithm 2 describes the process of pruning isolated control dependence subgraphs. It first generates a set of control dependence subgraphs. For each subgraph  $g$ , it records the number of dependencies with the statements outside  $g$  (lines 2–6). It then only keeps isolated control dependence subgraphs in  $CG$  (line 7). The algorithm keeps removing remaining subgraphs in  $CG$  until the number of removed nodes is greater than a given threshold  $|V(G)| \times \lambda$  (lines 8–14). It finally returns the new graph as well as the number of removed nodes.

**4.3. Reconstructing Dependence Subgraphs.** We can see from the three plagiarist programs  $P_2$ ,  $P_3$ , and  $P_4$  in Figures 4(b)–4(d) that the data dependencies are hidden by the confusing variables and statements inserted by copyists. To retain the execution of the source program, the inserted statements have to deliver such data dependencies. As their corresponding PDGs  $G(P_2)$ ,  $G(P_3)$ , and  $G(P_4)$  in Figures 4(f)–4(h) show, they all include a common path from the lowest node to the highest. Compared with the PDG  $G(P_1)$  in Figure 4(e), these four PDGs are unlikely to be judged as isomorphic, and therefore it is difficult for previous detection tools or algorithms to detect such plagiarism method.

In order to deal with this antidetection method, we propose a control subgraph reconstruction algorithm, since reconstruction of control subgraph can eliminate the influences on PDG brought by most inserted variables and

statements and thereby retain the PDGs of source program and plagiaristic program, largely increasing the algorithm's detection ability. As Figure 4 shows, to break original dependencies, copyists insert confusing variables and related operation statements, but on the other hand, they have to indirectly express those dependencies to make sure the execution results remain the same. Therefore the goal of reconstructing control subgraph is to restore those dependencies as much as possible or to express them with the same pattern. In this way, the PDGs of original program and plagiaristic program are the same. To achieve this goal, we have to firstly define some critical statements for expressing dependence. The basic idea of control subgraph reconstruction algorithm is uniformly expressing dependencies using these critical statements.

Since dependencies are to be transformed to dependencies of critical statements, how to select critical statements is thusly important. Critical statements commonly contain much semantics. Usually copyists conservatively add confusing variables and statements to ensure that the execution semantics are not changed. Obviously these added statements should not be critical statements, which means the program's execution semantics should not be expressed by these statements.

**Definition 3** (critical statement). A program statement is a critical statement, if it satisfies one of the following conditions:

- (i) containing subprogram calls,
- (ii) reading or modifying nonlocal variables,
- (iii) containing return statements.

For example, statements in lines 4, 5, 11, 13, and 19 in Figure 1 are critical statements, which are more difficult to imitate compared with other statements, and their mutual relations are difficult for copyists to conceal.

```

Input: PDG  $G$  whose partial control subgraph has been deleted;
Output: PDG after reconstruction of control subgraph;
(1) Generate  $G$ 's control dependence subgraph set  $CG = \{g_1, \dots, g_k\}$ ;
(2) foreach control dependence subgraph  $g$  in  $CG$  do
(3)   Mark irremovable nodes in  $g$ ;
(4)   foreach irremovable node  $S$  in  $g$  do
(5)     Build an empty irremovable node Set;
(6)     Build a queue  $Q$  and put nodes that  $S$  is data-dependent on into  $Q$ ;
(7)     while  $Q$  is not empty do
(8)        $P = Q.dequeue()$ ;
(9)       if  $P$  is irremovable node then
(10)        Put  $P$  into Set;
(11)      else
(12)        Put nodes that  $P$  is data-dependent on into  $Q$ ;
(13)      Add  $S$  into each node's data dependence in Set;
(14) Delete all removable nodes and their dependencies in  $g$ ;
(15) return  $G$ ;

```

ALGORITHM 3: Reconstruction of control dependence subgraph.

Critical statement evaluates the importance of statement only from the point of program, while, analyzed from the point of PDG, another kind of statement is also very important, which is external-dependent statement. Every noncontrol statement belongs to some certain control dependence graph, and there exists such statement  $S$  that comes from statements in other PDGs or data dependencies in some control statements. Such statement  $S$  is called external-dependent statement, which is important because the values of variables it modifies are used by some loops or select statements. In this case, the execution path can only be determined when program is running, and therefore copyists have to retain this external dependence to ensure the correctness of program. Therefore we also use external-dependent statements as well as critical statements to express dependence in this paper.

The goal of control subgraph reconstruction algorithm is reexpressing the dependencies within statements in control subgraphs, expressing them with uniform statements. Therefore this algorithm can eliminate most disguised dependencies. To elaborate this algorithm, irremovable node is defined in Definition 4.

*Definition 4* (irremovable node). The nodes for all critical statements and external-dependent statements in PDG  $G$  are irremovable nodes, and the others are removable nodes.

The main processes of control subgraph reconstruction algorithm are dealing with every control subgraph, deleting all removable nodes in subgraphs and transforming the dependencies of irremovable nodes on removable nodes into dependence on irremovable nodes dependent on these removable nodes. Take the programs in Figures 4(a)–4(d) as examples, statement  $findVal = temp/2$  is an external-dependent statement (because control statement  $inputVal == findVal$  is data-dependent on this statement), and therefore it is an irremovable node. However, this

statement is dependent on removable nodes in all these four programs. For example, node 7 in Figure 4(e) is a removable node. By using such nodes, copyists disguise the dependence in original program, and our algorithm will remove these nodes and meanwhile transform the dependence of  $findVal = temp/2$  on removable nodes into dependence on irremovable nodes that the removable nodes depend on. All the gray nodes in  $G(P_1)$ ,  $G(P_2)$ ,  $G(P_3)$ , and  $G(P_4)$  are irremovable nodes and white nodes are removable nodes. Our algorithm will remove removable nodes and keep the dependencies among irremovable nodes. We show that the graphs for plagiaristic programs and original programs are the same after reconstruction. One principle for control subgraph reconstruction is retaining the PDG structure of original program as much as possible and eliminating the influences of copyists on them.

Algorithm 3 describes this algorithm. The loop from line 2 to line 14 deals with one control dependence subgraph each time. The algorithm marks the critical nodes and external-dependent statements in every control subgraph (line 3). The loop from line 4 to line 13 deals with all irremovable nodes in a control subgraph, while lines 7–12 are the breadth first search on subgraph  $g$ , where the search stops when encountering irremovable nodes (lines 9–10). If encountering removable nodes, the nodes they depend on are added to queue (line 12). This loop continues until the queue  $Q$  is empty which means statement  $S$  no longer depends on any removable nodes. It adds  $S$  to the irremovable nodes set (line 13). In line 14, all removable nodes and their corresponding dependence are deleted. The four PDGs  $G'(P_1)$ ,  $G'(P_2)$ ,  $G'(P_3)$ , and  $G'(P_4)$  are the reconstructed subgraphs of  $G(P_1)$ ,  $G(P_2)$ ,  $G(P_3)$ , and  $G(P_4)$ , respectively.

After pruning isolated control dependence subgraphs from a PDG, and reconstructing PDGs, some unique subgraph patterns could be removed from the PDG, therefore shrinking the number of dimensions in the feature space  $\mathcal{T}$ .

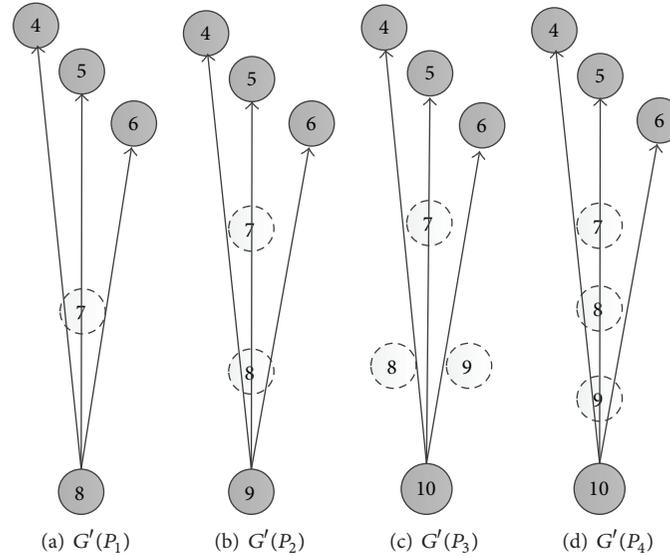


FIGURE 5: The PDGs after reconstruction.

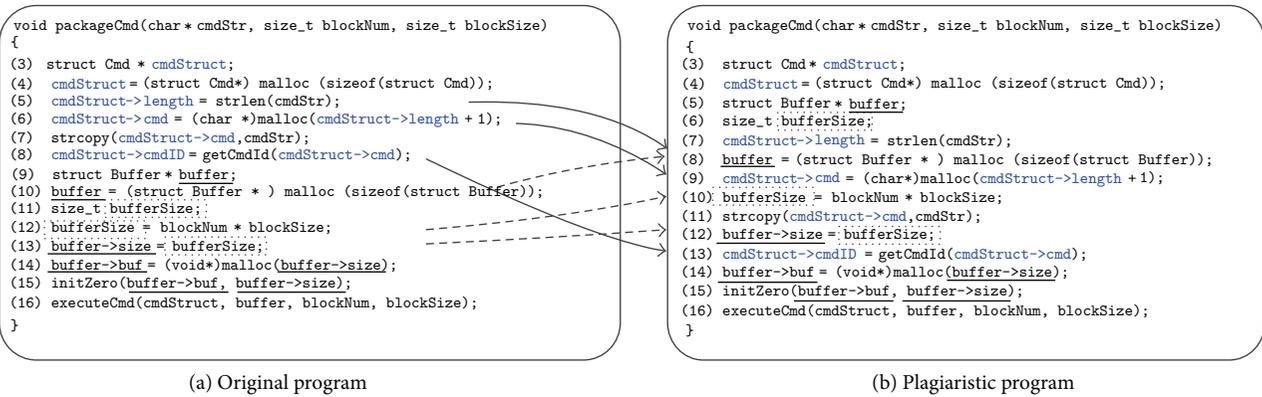


FIGURE 6: An example of changing the order of statements.

## 5. Determining Copy Direction

Besides inserting variables or statements in a program  $P_i$ , copyists would also modify the sequence of statements (on the premise that the execution semantics remain the same) to escape the detection of some algorithms based on statements or token sequence. In this section we propose a PDG reconstruction approach by considering the order of statements (Figure 5).

*5.1. Effect of Data Dependencies.* Generally, the styles and features of code have some fixed regulations. As for the example in Figure 6 two programs have exactly the same execution semantics. The only difference is the order of their statements. For example, the statements in lines 3–8 are related to the variable `cmdStruct` and they are consecutive. While in Figure 6(b), these statements are disperse, with other irrelevant statements mixed in. Obviously, relevant statements are always put together when programming, and therefore the code in Figure 6(a) conforms to programmers'

logic better because consecutive statements have stronger relevance.

We summarize the following two features of statements in a source program.

*Feature 2.* In a source program  $P_i$ , variables in the same structure should be modified by successive statements.

For instance, statements in lines 4–6 in Figure 6(a) are all assignments of a structure variable `cmdStruct`, and therefore they are highly relevant. While in Figure 6(b), they are scattered in lines 4, 7, and 9, respectively. As a result, the program in Figure 6(b) is more likely to be a plagiaristic program.

*Feature 3.* In a source program  $P_i$ , on average, every two successive statements might be more data-dependent than they are in a plagiaristic program.

For example, in Figure 6(a), programmers are likely to consecutively write down statements related to `bufferSize` in lines 12–14, and these statements are expected to be in

consecutive order. However, they are dispersed in lines 10, 12, and 14 in Figure 6(b).

Given any two statements  $S_i$  and  $S_{i+1}$ , we define two scores  $I_1(S_i + S_{i+1})$  and  $I_2(S_i + S_{i+1})$  to measure their level of relevance. The higher the scores are, the more likely their serial numbers are consecutive:

$$I_1(S_i + S_{i+1}) = \begin{cases} 1 & \text{if } S_i \text{ and } S_{i+1} \text{ assign the same data structure,} \\ 0 & \text{otherwise.} \end{cases}$$

$$I_2(S_i + S_{i+1}) = \begin{cases} 1 & \text{if } S_{i+1} \text{ is data-dependent on } S_i, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Therefore, given a program, if most consecutive statements in a program  $P$  are not data-dependent on each other, we will give a relatively high score to this program  $P$ . We accumulate these two scores for every two statements and get the total score

$$I(P) = \frac{n-1}{\sum_{i=1}^{n-1} (I_1(S_i + S_{i+1}) + I_2(S_i + S_{i+1}))}, \quad (7)$$

where  $S_i, S_{i+1} \in P$  and  $n$  is the number of statements in  $P$ .

**5.2. Calculating Copy Degree by Combining Data Dependence and Control Dependence.** We determine copy direction based on the following three factors by considering data dependencies and control dependencies in a PDG.

- (i) Number of isolated control subgraphs in  $G(P)$ , denoted by  $N_g(P)$ : the larger it is, the more ‘‘bloated’’ the program is. This is probably caused by code fragment inserted by copyists to avoid being detected, and therefore a greater  $N_g(P)$  indicates a greater possibility for a program to be plagiaristic.
- (ii) Number of removable nodes in  $G(P)$ , denoted by  $N_n(P)$ : removable nodes represent noncritical statements and statements without any external dependence. The analysis in Section 4.3 shows that most statements inserted by copyists are removable nodes, and therefore the bigger  $N_n$  is, the more likely the program is plagiaristic.
- (iii) Data dependence score  $I(P)$ ; a greater  $I(P)$  indicates a greater possibility for a program to be plagiaristic.

Let  $P_i$  and  $P_j$  be two programs, and let  $G'(P_i)$  and  $G'(P_j)$  be two similar reconstructed PDGs determined by the ELM. Equation (8) shows the copy degree of  $P_i \rightsquigarrow P_j$ . We say  $P_i \rightsquigarrow P_j$  if  $\text{score}(P_i \rightsquigarrow P_j) < 1$ , otherwise  $P_i$  copies  $P_j$  if  $\text{score}(P_i \rightsquigarrow P_j) > 1$ :

$$\text{score}(P_i \rightsquigarrow P_j) = \frac{N_g(P_i) + 1}{N_g(P_j) + 1} \times \frac{N_n(P_i) + 1}{N_n(P_j) + 1} \times \frac{I(P_i)}{I(P_j)}. \quad (8)$$

## 6. Experiments

All the algorithms were implemented using GNU C++. The experiments were run on a PC with an Intel(R) Core(TM)2 Duo CPU T6600@2.20 GHz, 2 GB RAM, running a Ubuntu (Linux) 32-bit operating system. We adopted Frama-c as a source code analysis tool to generate PDGs.

We ran our experiments on C program sets collected from the Internet. To ensure the validity of algorithm, we copied part of the functions in these programs and adopted the following plagiarism methods [11] to disguise them as follows.

- (i) Whenever  $m$  (usually 2 to 4) consecutive statements are not bounded by dependencies, reorder them.
- (ii) Replace a `while` loop with an equivalent `for` loop and vice versa. Occasionally, a `for` loop is replaced by an infinite `while` loop with a `break` statement.
- (iii) Scatter variables in the same structure and keep their dependencies.
- (iv) Replace `if (a) A` with `if (! (!a)) A` and `if (a) A else B` with `if (!a) B else A`; recurse `if` nested `if` block is encountered. Finally, apply DeMorgan’s rule if the boolean expression is complex.
- (v) Run JPLAG. For any places that they are not confused, insert a statement or a label.
- (vi) Run test scripts, and ensure that correctness is preserved during plagiarism.

We then put plagiarist codes together with original program sets to get test data set, including 250 functions and nearly 4000 lines of program. We labeled source and plagiaristic functions for evaluating detecting algorithms.

We compare our algorithms with the following three state-of-the-art tools:

- (i) GPLAG: a classic source code plagiarism detection software [11],
- (ii) JPLAG: a tool of finding plagiarisms among a set of programs [12],
- (iii) PlaGate: a tool of integrating with existing plagiarism detection tools to improve plagiarism detection performance [13].

Notice that, to our best knowledge, there is no report on detecting copy directions among programs. So we compared our approaches with GPLAG, JPLAG, and PlaGate to evaluate the precision and recall of finding plagiarist programs without detecting copy directions. We set support threshold  $\rho = 3$  and the percentage of removed isolated control dependence subgraphs  $\lambda = 20\%$ .

**6.1. Accuracy of Detecting Plagiarisms.** In order to compare with other tools, we used precision and recall for copy relationship to evaluate accuracy of detecting plagiarisms.

*Precision for copy relationship* is the percentage of correctly detected source-plagiarism pairs in the list of pairs detected using detection algorithm. Precision is 1 when every pair detected is a source-plagiarism pair.

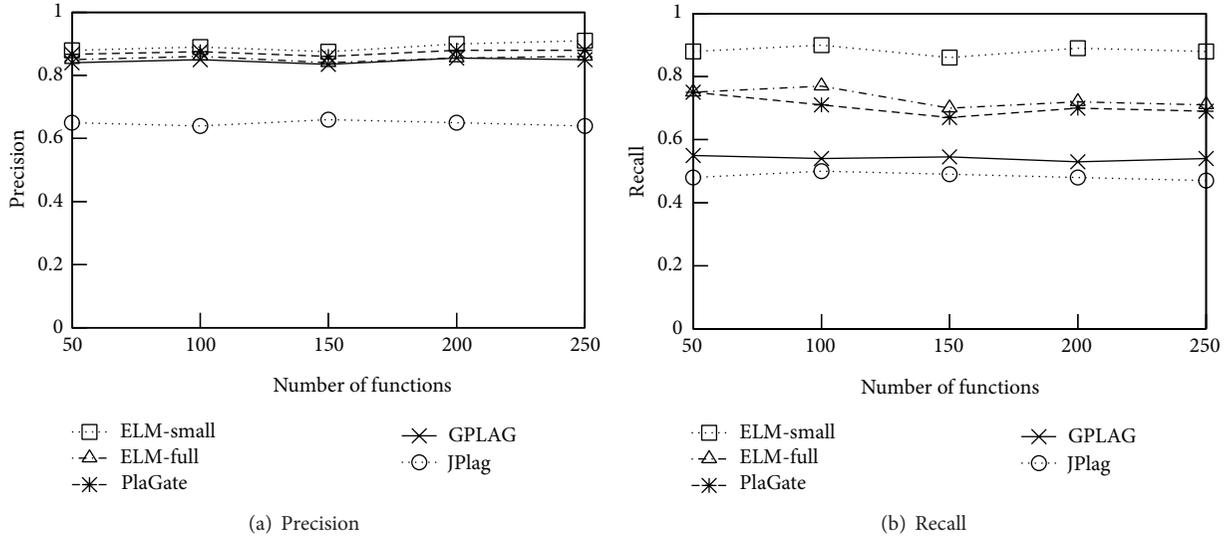


FIGURE 7: Comparison of algorithms.

*Recall for copy relationship* is the ratio percentage source-plagiarism pairs that are detected to all labeled source-plagiarism pairs. Recall is 1 when all source-plagiarism pairs are detected.

Figure 7(a) shows precision of different detection approaches. ELM-full represents the algorithm using full feature space and ELM-small represents the algorithm using pruned feature space. Both ELM-based approaches had a higher accuracy than GPLAG. Since GPLAG is also based on PDG, the results show that PDG-based approaches can reflect programs' semantical features well. JPLAG, which is based on sequence, was influenced by some small functions, leading to the decrease of accuracy. PlaGate, which is based on latent semantic analysis, was very close to our ELM-full.

As for recall, Figure 7(b) shows that ELM-small has a recall of nearly 90%, ELM-full has a recall of 77%, while GPLAG, which is also based on PDG, has a recall of only 55%. This is because many plagiarisms adopt methods to disguise codes, which greatly changed the structure of PDG, making GPLAG unable to detect such plagiarism. JPLAG could only detect plagiarism without any modification, having an accuracy of around 50%, coinciding with the fact that half of the plagiarism was not disguised. PlaGate integrated existing plagiarism detection tools to improve plagiarism detection performance, which was higher than the other state-of-the-art tools, but less than ELM-small since the existing approaches did not consider removing and reconstruction PDGs.

**6.2. Accuracy of Determining Copy Direction.** We further evaluated accuracy of determining copy directions and defined *precision for copy directions* as the percentage of correctly detected plagiarism program in the list of all detected plagiarisms using detection algorithm. Precision is 1 when every function detected is a plagiarism.

Figure 8 shows the accuracy of determining copy direction. Because no previous work can figure out plagiarism

direction between programs, we only report detection results of using our ELM based approaches. Among 250 functions, 65 plagiarist functions were exactly the same with source functions.

Figure 8(a) shows the precision when the data set contains 65 unchanged plagiaristic programs, and Figure 8(b) shows the precision when the data set does not contain any unchanged plagiaristic functions. The precision shown in Figure 8(a) is only around 75%, since our approaches could not figure out the copy direction when plagiaristic programs were exactly the same with source programs. The precision in Figure 8(b) is around 95%, which shows that our approaches can distinguish the copy direction and further detect most plagiarisms.

The recall ratio was the same with the results shown in Figure 7(b) since among all labeled plagiarisms, the number of detected plagiarisms was the same with the number of detected copy-plagiarism pairs and the number of labeled plagiarisms was the same with the number of labeled source-plagiarism pairs.

**6.3. Detection Time.** We compared time for detecting programs with plagiaristic relationship. Figure 9(a) shows the comparison results. The detecting time mainly depended on the time cost for static analysis tools and the efficiency of identifying programs with plagiaristic relationship. Our approaches and GPLAG costed the same time for static analysis, however, by using ELM, our ELM-based approach ran faster than GPLAG since GPLAG needed to find isomorphic subgraphs, which was time consuming, whereas we used ELM to classify PDGs, which was very efficient. PlaGate ran slowest since it required more detection time to integrate existing plagiarism detecting tools [13].

We also test the time for detecting copy directions. Since our work is the first one to detect copy directions, we only report the detecting time of our approaches using different feature spaces. Figure 9(b) shows the running time for

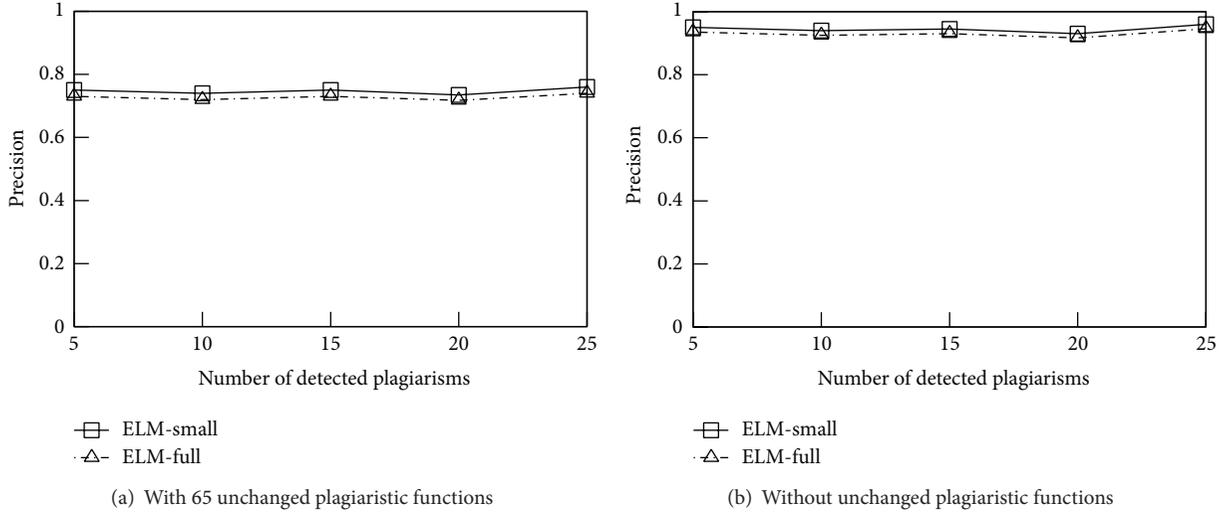


FIGURE 8: Precision of determining copy direction.

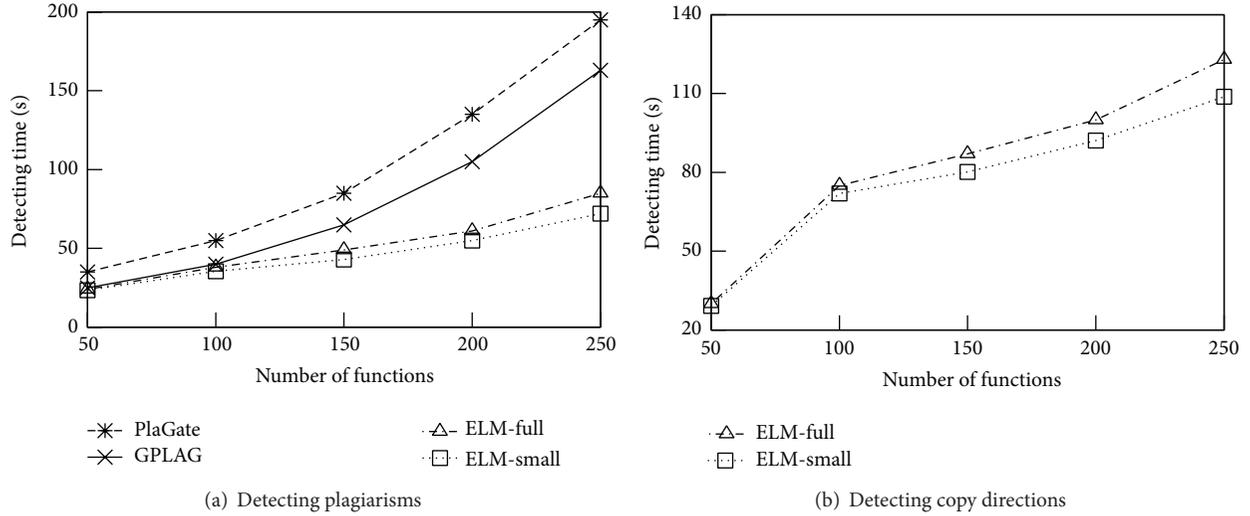


FIGURE 9: Detecting time.

detecting copy directions. Besides identifying programs with plagiaristic relationship, our approaches needed time to detect copy directions by considering data dependencies and control dependencies in PDGs.

Figure 10 shows number of generated feature spaces for different number of functions. We can see that pruning isolated control dependence subgraphs decreased relatively high number of features in each feature space.

**6.4. Effect of Feature Space Size.** Figure 12 shows the effect of support threshold  $\rho$  on our algorithms. We used the whole data set with 250 functions,  $\lambda = 20\%$ , and varied  $\rho$  from 3 to 15.

Figure 11(a) shows that the size of feature space  $\mathcal{T}$  shrinks when increasing  $\rho$ ; particularly when  $\rho$  increases to 9, size  $|\mathcal{T}|$  drops quickly. It is consistent with our analysis in Section 3. The detection time dropped correspondingly as shown in

Figure 11(b). It is interesting to see that ELM-small improved the detection time when  $\rho$  was small but costed a lot when  $\rho$  increased to 12. This is because ELM-small required time to remove isolated control dependence subgraphs. When  $\rho$  was small, removing such subgraphs improved detection time a lot; however, when  $\rho$  was large, removing subgraphs from a small feature space would not bring benefit.

Figures 11(c) and 11(d) show that when increasing  $\rho$ , precision of the two algorithms drops slightly and recall drops significantly. Using a small  $\rho$ , our approaches chose large subgraphs in PDGs as patterns, which contain representative features in  $\mathcal{T}$ . So ELM-full provided relatively high precision and recall ratios. Based on the large feature space, ELM-small removed isolated control dependence subgraphs to further enhance the accuracy. When increasing  $\rho$ , the number of detected similar subgraphs in every PDGs pairs decreased, so recall dropped quickly. However, the precision in the detected

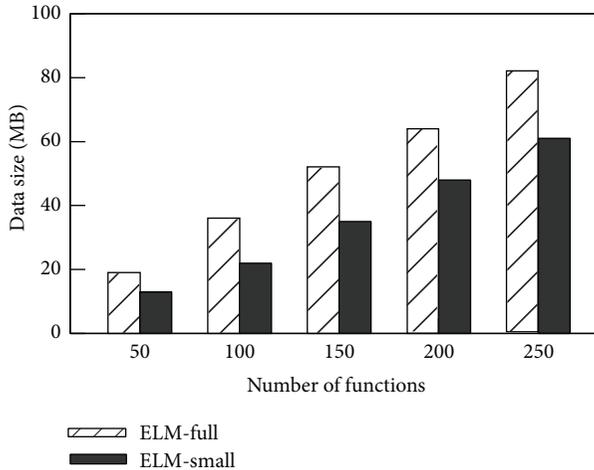


FIGURE 10: Feature space.

pairs did not change too much. These figures also show that the effect of removing isolated control dependence subgraphs disappears gradually when  $\rho$  becomes large since a smaller feature space corresponds to less isolated control dependence subgraphs.

**6.5. Effect of Removed Isolated Control Dependence Subgraphs.** Figure 12 shows the effect of parameter  $\lambda$  on ELM-small algorithm. We used the whole data set with 250 functions,  $\rho = 3$ , and varied  $\lambda$  from 5 to 30. Figure 12(a) shows that precision drops slightly but is not affected too much when increasing  $\lambda$ . Figure 12(a) shows that recall increases quickly when increasing  $\lambda$  from 5% to 20% and then increases slowly, which indicates that removing partial isolated control dependence subgraphs will help to detect more source-plagiarism pairs. And it is not surprising to see removing more isolated control dependence subgraphs makes detection faster as shown in Figure 12(c).

## 7. Related Work

The detection of source code plagiarism basically contains three steps: *process*, *transform*, and *match* [11, 14, 15]. The operations in *process* are dealing with original source code, extracting information that would be used in subsequent operations and removing unnecessary information for judging similarity. For example, most algorithms would remove blank characters and comments in *process*. The second step is *transform*. In *transform*, preprocessed programs will be transformed into certain intermediate form which varies according to algorithm. Programs are represented by intermediate form because it only contains information needed for detecting similarity and is beneficial to subsequent calculations in *match*, which only needs intermediate form to determine whether code fragments are similar. The intermediate form of an algorithm is important, because it determines the basis for judging similarity between source codes, and the effectiveness of an algorithm depends on whether its intermediate form can completely and correctly represent the

similarity relation between original codes. Meanwhile intermediate form also determines the algorithm's efficiency. For instance, an algorithm adopting a tree-based intermediate form generally runs faster than a graph-based one but has lower correctness. The next step is *match*, which is conducting similar pair operations on intermediate form of the source program. This is the key step in the entire algorithm.

Based on different intermediate form, the existing work can generally be categorized into five groups.

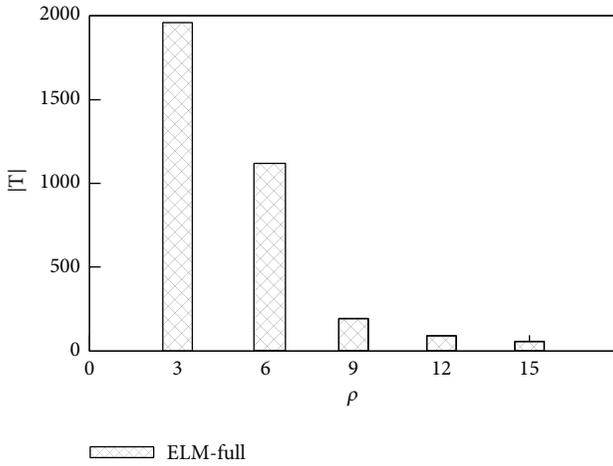
(i) *Text-Based Method*. Since source code is a text set, text-based methods directly use source code as intermediate form to compare without utilizing the grammar and syntax features of program. Johnson first proposed a text-based method. In [16, 17], he firstly separated the source code into several equi-length sub-programs, and then conducted hash calculations on separated programs. Each hash value is called "fingerprint." In *match*, a slide window was used to recognize program fragments with the same hash value. To recognize fragments of different lengths, the slide window algorithm needs running several times to find similar programs of different lengths.

Ducasse et al. in [18] proposed another text-based method, based on an idea of using a two-dimension axis, where each axis represents a set of source codes, and each point on the axis is a row of source code. The value of the point is the hash value of this source code. If a point  $A$  on  $X$  axis has the same value with a point  $B$  on  $Y$  axis, then the point  $(A, B)$  will be marked. Therefore the problem of detecting similar code fragments can be transformed into finding diagonal connected points. Ducasse used a pattern matching algorithm to find diagonal connected points, which allows a minor number of point misses.

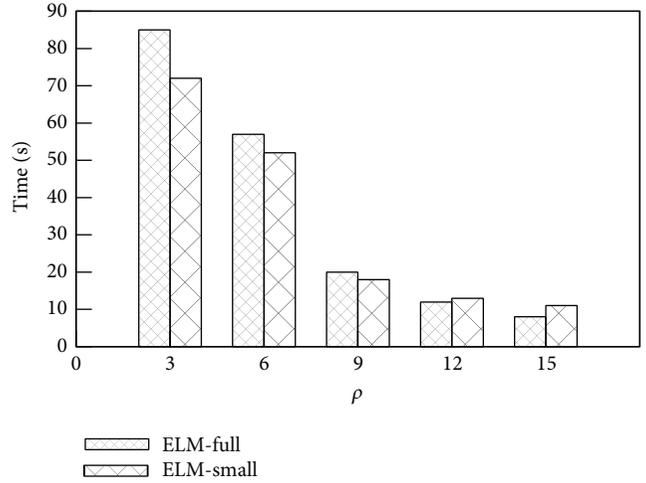
(ii) *Word-Based Method*. Word-based methods lexically analyze source code at first and transform source code into a sequence consisting of token, on which the subsequent steps will be conducted. Obviously, word-based methods can better deal with modifications on code fragments like variable rename, pattern adjustment, and so on. Compared with text-based methods, they are more robust.

There are many word-based methods, one of which is CCFinder proposed by Kamiya et al. in [15]. It firstly formalizes the source program, which is aimed at removing incorrect influences of matching results caused by some languages' grammar. Take scope operator in C++ language as an example; it has no meaning in judging similarity, and as a result it can be removed during formalization. Moving on, CCFinder would conduct lexical analysis to programs to transform the program into a parameterized token sequence. Its matching operation is using this sequence to build a suffix tree, which is used for figuring out identical subsequences as similar program fragments.

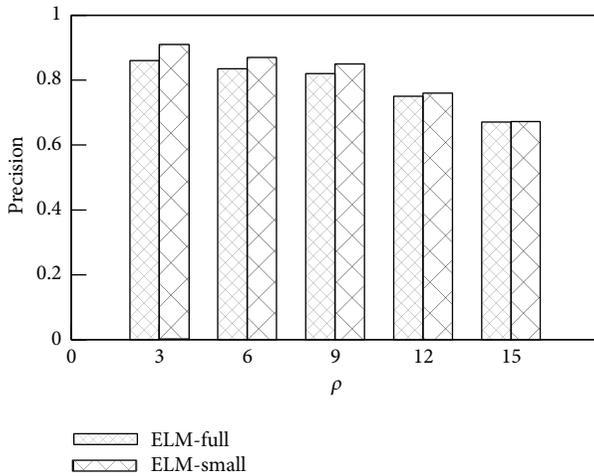
CP-Miner is another representative algorithm, which divides program and then calculate every row's hash value. Afterward programs are transformed to a sequence of hash value. Before matching, CP-Miner firstly divides hash value sequence into several subsequences according to program's scope. Thus programs are expressed as a set of subsequences.



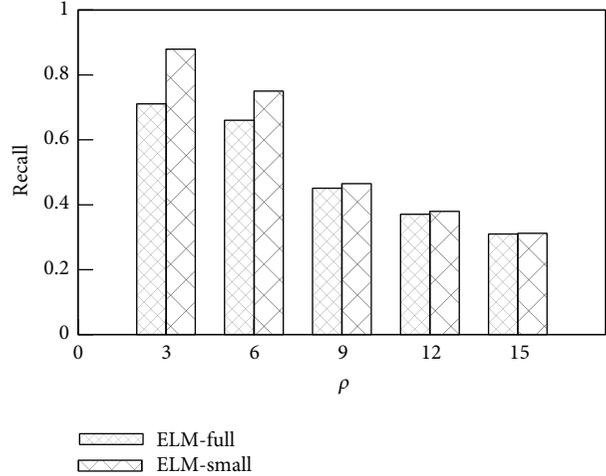
(a)  $\mathcal{T}$  versus  $\rho$



(b) Detecting plagiarisms time



(c) Precision versus  $\rho$



(d) Recall versus  $\rho$

FIGURE 11: Effect of parameter  $\rho$ .

CP-Miner adopts CloSpan [19], a frequent subsequence algorithm, to figure out subsequences which appear frequently in programs' subsequence set. These subsequences are considered as similar programs.

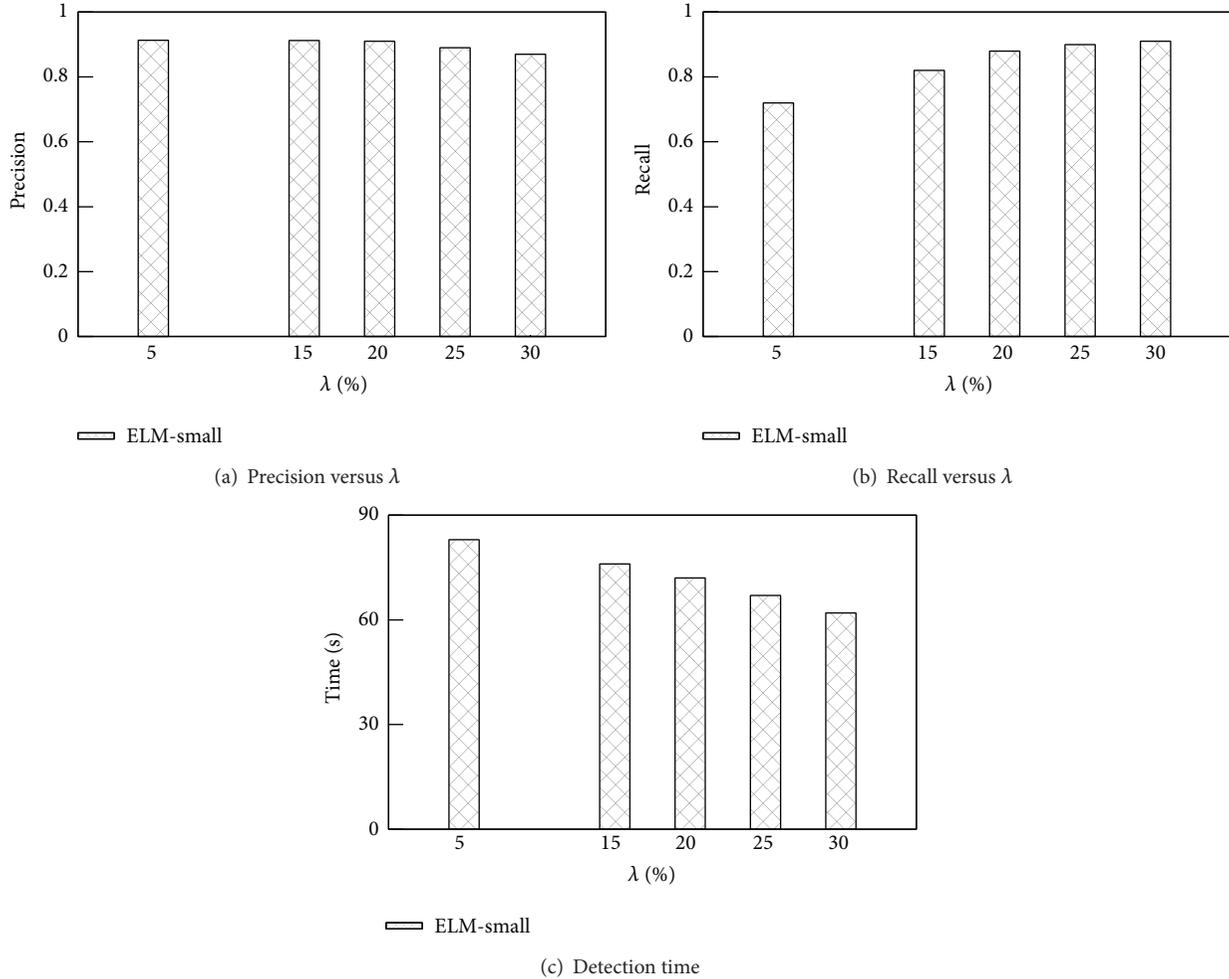
(iii) *Tree-Based Method.* Since program is a kind of text with precise meaning, detecting similar programs can be more accurately completed by further analyzing the programs. To begin with, tree-based methods conduct lexical and grammatical analysis on the program, transform it into an abstract syntax tree, and then use tree matching or other methods to conduct similarity detection.

Baxter et al. in [14] proposed a tree-based method CloneDr, which uses a compiler generator to generate a commented syntax tree generator. Then a tree matching algorithm is used to do one on one matching. To reduce running time, an optional procedure is to conduct hash operations on generated subtrees and then compare subtrees with the same hash value pairwise. By doing this, running time can be largely reduced.

The time complexity of subtree comparison is large, and to avoid subtree matching, Koschke et al. in [20, 21] built an abstract syntax tree for the program first of all and then sorted them, transforming the tree into a sequence of tree nodes to build a suffix tree for similarity matching.

Jiang et al. in [22] proposed a novel algorithm Deckard. Similarly, Deckard firstly computes abstract syntax tree in programs and calculates subtrees' eigenvector which is used to approximately represent trees. After that, LSH technology is used to cluster eigenvectors to find similar program fragments.

(iv) *Metric-Based Method.* The basic idea of metric-based methods is to extract some features or indicators from program fragments and thereby use them to compare instead of directly comparing code or abstract syntax trees. A prevalent method is to extract some parameters as fingerprints or features from every program's grammar unit, such as class, function, method, and statement. To begin with, most algorithms would grammatically analyze the programs to

FIGURE 12: Effect of  $\lambda$ .

compute its abstract syntax tree or PDG, based on which eigenvalues are computed. Huang et al. in [9] adopted some features to represent functions, and functions with the same eigenvalue are recognized as similar programs. The calculations of these eigenvalues come from functions' names, distributions, expressions, and their simple control flows.

(v) *Semantic-Based Method.* Semantic-based methods are used to find more veiled similarities among programs. The source code of plagiaristic programs may be quite different but must be semantically similar with copied programs. Therefore semantic-based methods conduct static analysis instead of simple grammatical analysis to provide more precise information. Many semantic-based methods adopt PDG to represent source programs. PDG is a graph structure that represents the inner data and control dependence of the program, where the nodes are used to represent programs' statements, such as statements and expressions. By using such representation, consecutive statements that have no semantic relation can be independent. Therefore the problem of finding similar programs is transformed into the problem of finding isomorphic subgraphs. Komondoor and Horwitz in

[23] proposed a PDG-based method, using reverse program fragments to find isomorphic subgraphs. Liu et al. in [11] developed a PDG-based tool GPLAG to detect plagiarism.

There are other methods using semantic analysis to detect similar program fragments except PDG-based ones. Kim et al. in [24] proposed a method based on static memory comparison. Its basic idea is to use static semantic analyzer to build an abstract memory status for every subprogram and compute the similarity between two programs by comparing the similarity of memory status. This method is completely based on semantics that even if two programs are literally different, their abstract memory can be similar. And similar memory status reflects programs' similar functions.

## 8. Conclusion and Future Work

Aimed at the disadvantages of current source code plagiarism detection tools and algorithms, we propose a new ELM-based detection approach in this paper. We summarize common methods of plagiarism, deeply analyze the effects of these methods on PDGs, and propose corresponding detection strategies. Our detection approach can detect plagiarism

between source codes that most classic detection algorithms cannot detect, and its correctness has been proved.

Our future work will focus on how to analyze program's semantics and thereby improve detection ability of algorithm.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The work is partially supported by the National Natural Science Foundation of China (no. 61272178), the National Basic Research Program of China (973 Program) (no. 2012CB316201), the National Natural Science Foundation of China (nos. 61322208, 61129002), the Doctoral Fund of Ministry of Education of China (no. 20110042110028), and the Fundamental Research Funds for the Central Universities (no. N110804002).

## References

- [1] J. Ferrante, K. J. Ottenstein, and J. D. Warren, "The program dependence graph and its use in optimization," *ACM Transactions on Programming Languages and Systems*, vol. 9, no. 3, pp. 319–349, 1987.
- [2] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [3] G.-B. Huang, C.-K. Siew, and L. Chen, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [4] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on bow framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, June 2014.
- [5] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.
- [6] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.
- [7] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [8] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.
- [9] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [10] X. Yan and J. Han, "CloseGraph: mining closed frequent graph patterns," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 286–295, Washington, DC, USA, August 2003.
- [11] C. Liu, C. Chen, J. Han, and P. S. Yu, "GPLAG: detection of software plagiarism by program dependence graph analysis," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*, pp. 872–881, August 2006.
- [12] L. Prechelt, G. Malpohl, and M. Philippsen, "Finding plagiarisms among a set of programs with JPlag," *Journal of Universal Computer Science*, vol. 8, no. 11, pp. 1016–1038, 2002.
- [13] G. Cosma and M. Joy, "An approach to source-code plagiarism detection and investigation using latent semantic analysis," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 379–394, 2012.
- [14] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM '98)*, pp. 368–377, Bethesda, Md, USA, November 1998.
- [15] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilingual token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 654–670, 2002.
- [16] J. Johnson, "Identifying redundancy in source code using fingerprints," in *Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON '93)*, pp. 171–183, 1993.
- [17] J. Johnson, "Visualizing textual redundancy in legacy source," in *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON '94)*, p. 32, 1994.
- [18] S. Ducasse, M. Rieger, and S. Demeyer, "A language independent approach for detecting duplicated code," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM '99)*, pp. 109–118, September 1999.
- [19] X. Yan, J. Han, and R. Afshar, "Clospan, Mining closed sequential patterns in large databases," in *Proceedings of the 3rd SIAM International Conference on Data Mining*, pp. 166–177, San Francisco, Calif, USA, May 2003.
- [20] R. Falke, P. Frenzel, and R. Koschke, "Empirical evaluation of clone detection using syntax suffix trees," *Empirical Software Engineering*, vol. 13, no. 6, pp. 601–643, 2008.
- [21] R. Koschke, R. Falke, and P. Frenzel, "Clone detection using abstract syntax suffix trees," in *Proceedings of the 13th Working Conference on Reverse Engineering (WCRE '06)*, pp. 253–262, October 2006.
- [22] L. Jiang, G. Mishnerghi, Z. Su, and S. Glondu, "DECKARD: scalable and accurate tree-based detection of code clones," in *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*, pp. 96–105, Minneapolis, Minn, USA, May 2007.
- [23] R. Komondoor and S. Horwitz, "Using slicing to identify duplication in source code," in *Proceedings of the 8th International Symposium on Static Analysis (SAS '01)*, pp. 40–56, 2001.
- [24] H. Kim, Y. Jung, S. Kim, and K. Yi, "MeCC: memory comparison-based clone detector," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*, pp. 301–310, May 2011.

## Research Article

# Online Sequential Prediction for Nonstationary Time Series with New Weight-Setting Strategy Using Extreme Learning Machine

Wentao Mao,<sup>1</sup> Jinwan Wang,<sup>1</sup> Liyun Wang,<sup>1</sup> and Mei Tian<sup>2</sup>

<sup>1</sup>College of Computer and Information Engineering, Henan Normal University, Henan, Xinxiang 453007, China

<sup>2</sup>Management Institute, Xinxiang Medical University, Henan, Xinxiang 453003, China

Correspondence should be addressed to Wentao Mao; maowt.mail@gmail.com

Received 21 August 2014; Accepted 12 October 2014

Academic Editor: Amaury Lendasse

Copyright © 2015 Wentao Mao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Accurate and fast prediction of nonstationary time series is challenging and of great interest in both practical and academic areas. In this paper, an online sequential extreme learning machine with new weighted strategy is proposed for nonstationary time series prediction. First, a new leave-one-out (LOO) cross-validation error estimation for online sequential data is proposed based on inversion of block matrix. Second, a new weighted strategy based on the proposed LOO error estimation is proposed. This strategy ranks the samples' importance by means of the LOO error of each new added sample and then assigns various weights. Performance comparisons of the proposed method with other existing algorithms are presented based on chaotic and real-world nonstationary time series data. The results show that the proposed method outperforms the classical ELM and OS-ELM in terms of generalization performance and numerical stability.

## 1. Introduction

Time series prediction is generally playing an important role in many engineering fields, for example, dynamic mechanics, weather diagnostics, and so on. The key goal of time series prediction is to mine the inner regular patterns in time data in order to predict future data effectively [1]. Many traditional methods such as AR, ARMA, and ARIMA are well applied to solving stationary time series prediction. However, in practical applications, time series is almost nonstationary, which restricts the stationary methods above. From Takens' phase space delay reconstructing theory [2], this kind of data generally needs to reconstruct the phase space via delay coordinate at first. For example, in chaotic time series the  $m$ -dimensional vector is defined as follows:

$$x(t-1) = [x(t-\tau), x(t-2\tau), \dots, x(t-m\tau)], \quad (1)$$

where  $m$  is embedding dimension and  $\tau$  is delay constant. The prediction model can be described as  $x(t) = f(x(t-1))$ , where  $f$  is a nonlinear map. From this reconstruction,

the time correlation is transformed to spatial correlation. Then support vector machines (SVMs) [3, 4], neural networks (NNs) [5, 6], and other machine learning methods [7, 8] are successfully introduced to approximate the spatial correlation in nonstationary time series data.

Generally speaking, there are two main challenges for predicting nonstationary time series effectively. One is how to choose a proper baseline algorithm which should be computationally inexpensive and accurate enough. Another is how to distinguish the importance of different samples in time series. Different from SVMs and NNs, extreme learning machine (ELM), introduced by Huang et al. [9], has shown its very high learning speed and good generalization performance in solving many problems of regression estimate and pattern recognition [10, 11]. As a sequential modification of ELM, online sequential ELM (OS-ELM) proposed by Liang et al. [12] can learn data one-by-one or chunk-by-chunk. In many applications such as time-series forecasting, OS-ELMs also show good generalization at extremely fast learning speed. Therefore, OS-ELM is a proper solution for the first

challenge. Many researches were devoted to solve the second challenge. As recent data usually carry more important information than the distant past data, a typical and effective method is weight-setting. Lin and Wang [13] held the first sample in time series with the lowest importance while the most recent sample with the highest importance and then assigned fuzzy memberships to every sample. Tay and Cao [14] used exponential function to calculate every sample's importance in financial time series prediction. Very different from these stationary weight-setting strategy, Mao et al. [15] established a heuristic algorithm to dynamically choose the optimal weights. Bao et al. [16] solved this problem from multi-input multioutput perspective. He regarded the time samples as multiple outputs in a time slot, and utilized multidimensional SVM to establish model. Considering ELM, Wang and Han [17] introduced kernel trick on OS-ELM for nonstationary time series. Its essential idea is to transform spatial space for better approximation. Grigorievskiy et al. [18] used optimally-pruned ELM to tackle long-term time series prediction and obtained more comparable results than with SVM.

However, although ELM-based methods mentioned above work well in time series prediction [19], it still does not yet successfully solve the second challenge, that is, to distinguish different samples' significance. Specifically speaking, as a sample is added sequentially, it does not seem clear whether this sample is the most important or is even the newest. In this scenario, the inner structure hidden in time series data will determine the samples' significance, especially in nonstationary setting. In other words, it could not guarantee the new added sample to be most valuable for prediction. Therefore, to solve this problem, this paper firstly develops a new leave-one-out (LOO) cross-validation error estimation for OS-ELM aiming at time series prediction. Based on inversion of block matrix, this LOO estimation is fast enough for time series data. As proved by many theoretical works [20], LOO error using PRESS statistics is approximately unbiased and has been successfully applied to ELM with Tikhonov regularization [21]. To our best knowledge, this LOO error estimation is the first attempt to evaluate the generalization performance of OS-ELM on time series data. Moreover, this paper utilizes this LOO error estimation of each new added sample to measure its importance. Obeying this weight-setting strategy, this paper then proposes a new weighted learning method for OS-ELM. The short version of this paper has been published in the proceedings of 5th International Conference on Extreme Learning Machine (ELM 2014). Experimental results on chaotic and real-life time series data demonstrate the proposed method outperforms the traditional ELMs in generalization performance and numerical stable.

The paper is organized as follows. In Section 2, a brief review on OS-ELM and LOO cross-validation estimation is provided. In Section 3, we describe the LOO error estimation and the weighted learning algorithm of OS-ELM on time series data. Section 4 is devoted to computer experiments on two different types of time series data sets, followed by a conclusion of the paper in Section 5.

## 2. Brief Review

As the theoretical foundations of ELM, [22] studied the learning performance of SLFN on small-size data set and found that SLFN with at most  $N$  hidden neurons can learn  $N$  distinct samples with zero errors by adopting any bounded nonlinear activation function. Then, based on this concept, Huang et al. [9] pointed out that ELM can analytically determine the output weights by a simple matrix inversion procedure as soon as the input weights and hidden layer biases are generated randomly and then obtain good generalization performance with very high learning speed. Here a brief summary of ELM is provided.

Given a set of i.i.d. training samples  $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\} \subset \mathbf{R}^d \times \mathbf{R}^n$ , standard SLFNs with  $\tilde{N}$  hidden nodes are mathematically formulated as [9]:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \dots, N, \quad (2)$$

where  $g(x)$  is activation function,  $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{id}]^T$  is input weight vector connecting input nodes and the  $i$ th hidden node,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{in}]^T$  is the output weight vector connecting output nodes and the  $i$ th hidden node, and  $b_i$  is bias of the  $i$ th hidden node. Huang et al. [9] has rigorously proved that, then, for  $N$  arbitrary distinct samples and any  $(\mathbf{w}_i, b_i)$  randomly chosen from  $\mathbf{R}^d \times \mathbf{R}$  according to any continuous probability distribution, the hidden layer output matrix  $\mathbf{H}$  of a standard SLFN with  $N$  hidden nodes and is invertible and  $\|\mathbf{H}\beta - \mathbf{T}\| = 0$  with probability one if the activation function  $g: \mathbf{R} \mapsto \mathbf{R}$  is infinitely differentiable in any interval. Then, given  $(\mathbf{w}_i, b_i)$ , training a SLFN equals finding a least-squares solution of the following equation [9]:

$$\mathbf{H}\beta = \mathbf{T}, \quad (3)$$

where

$$\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathbf{x}_1, \dots, \mathbf{x}_{\tilde{N}}) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_{\tilde{N}} + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_{\tilde{N}} + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \quad (4)$$

$$\beta = [\beta_1, \dots, \beta_{\tilde{N}}]^T,$$

$$\mathbf{T} = [\mathbf{y}_1, \dots, \mathbf{y}_{\tilde{N}}]^T.$$

Considering most cases in which  $\tilde{N} \ll N$ ,  $\beta$  cannot be computed through the direct matrix inversion. Therefore, the smallest norm least-squares solution of (3) is calculated as follows:

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}, \quad (5)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ . Based on the analysis above, Huang et al. [9] proposed ELM whose framework can be stated as follows.

*Step 1.* Randomly generate input weight and bias  $(\mathbf{w}_i, b_i)$ ,  $i = 1, \dots, \bar{N}$ .

*Step 2.* Compute the hidden layer output matrix  $\mathbf{H}$ .

*Step 3.* Compute the output weight  $\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{T}$ .

Therefore, the output of SLFN can be calculated by  $(\mathbf{w}_i, b_i)$  and  $\hat{\boldsymbol{\beta}}$ :

$$f(\mathbf{x}_j) = \sum_{i=1}^{\bar{N}} \hat{\beta}_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \hat{\boldsymbol{\beta}} \cdot \mathbf{h}(\mathbf{x}_j). \quad (6)$$

Like ELM, all the hidden node parameters in OS-ELM are randomly generated, and the output weights are analytically determined based on the sequentially arrived data. OS-ELM process is divided into two steps: initialization phase and sequential learning phase [12].

*Step 1.* Initialization phase: choose a small chunk  $M_0 = \{(x_i, t_i), i = 1, 2, \dots, N_0\}$  of initial training data, where  $N_0 \geq \bar{N}$ .

- (1) Randomly generate the input weight  $\mathbf{w}_i$  and bias  $b_i$ ,  $i = 1, 2, \dots, \bar{N}$ . Calculate the initial hidden layer output matrix  $\mathbf{H}_0$ :

$$\begin{aligned} \mathbf{H}_0 &= \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \mathbf{h}(\mathbf{x}_2) \\ \vdots \\ \mathbf{h}(\mathbf{x}_{N_0}) \end{bmatrix} \\ &= \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\bar{N}} \cdot \mathbf{x}_1 + b_{\bar{N}}) \\ g(\mathbf{w}_1 \cdot \mathbf{x}_2 + b_1) & \cdots & g(\mathbf{w}_{\bar{N}} \cdot \mathbf{x}_2 + b_{\bar{N}}) \\ \vdots & & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_{N_0} + b_1) & \cdots & g(\mathbf{w}_{\bar{N}} \cdot \mathbf{x}_{N_0} + b_{\bar{N}}) \end{bmatrix}_{N_0 \times \bar{N}}. \end{aligned} \quad (7)$$

- (2) Calculate the output weight vector:

$$\boldsymbol{\beta}^0 = \mathbf{D}_0 \mathbf{H}_0^T \mathbf{T}_0, \quad (8)$$

where  $\mathbf{D}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$ ,  $\mathbf{T}_0 = [t_1, t_2, \dots, t_{N_0}]^T$ .

- (3) Set  $k = 0$ .

*Step 2.* Sequential learning phase.

- (1) Learn the  $(k + 1)$ th training data:  $d_{k+1} = (\mathbf{x}_{N_0+k+1}, t_{N_0+k+1})$ .
- (2) Calculate the partial hidden layer output matrix:

$$\begin{aligned} \mathbf{H}_{k+1} &= \left[ g(\mathbf{w}_1 \cdot \mathbf{x}_{N_0+k+1} + b_1) \cdots g(\mathbf{w}_L \cdot \mathbf{x}_{N_0+k+1} + b_L) \right]_{1 \times L}. \end{aligned} \quad (9)$$

Set  $\mathbf{T}_{k+1} = [t_{N_0+k+1}]^T$ .

- (3) Calculate the output weight vector:

$$\begin{aligned} \mathbf{D}_{k+1} &= \mathbf{D}_k - \mathbf{D}_k \mathbf{H}_{k+1}^T (\mathbf{I} + \mathbf{H}_{k+1} \mathbf{D}_k \mathbf{H}_{k+1}^T)^{-1} \mathbf{H}_{k+1} \mathbf{D}_k \\ \boldsymbol{\beta}^{k+1} &= \boldsymbol{\beta}^k + \mathbf{D}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^k). \end{aligned} \quad (10)$$

- (4) Set  $k = k + 1$ . Go to Step 2(1).

The generalization ability of ELM has been analyzed by many researchers. Lan et al. [23] added a refinement stage that used leave-one-out (LOO) error to evaluate the neurons significance in each backward step. Feng et al. [24] presented a fast LOO error estimation for regularized ELM. From the incremental learning point of view, Feng et al. [24] proposed an error minimized extreme learning machine which measured the residual error caused by adding a new added hidden node in an incremental manner. We highly recommend the following work. As for ELM, Liu et al. [25] derived a fast LOO error estimation of ELM. The generalization error in  $i$ th LOO iteration can be expressed as follows:

$$r_i = t_i - f_i(\mathbf{x}_i) = \frac{t_i - \mathbf{Hx}_i \mathbf{H}^\dagger \mathbf{T}}{1 - (\mathbf{Hx}_i \mathbf{H}^\dagger)_i}, \quad (11)$$

where  $(\cdot)_i$  means the  $i$ th element,  $\mathbf{H}$  is hidden layer matrix, and  $\mathbf{Hx}_i$  means the row about the sample  $\mathbf{x}_i$  in  $\mathbf{H}$ .

Liu et al. [25] have shown that the proposed algorithm can accurately calculate the LOO error and can avoid the  $N$  times observable model training process of the original cross-validation method. By the simulation experiment of artificial and real data sets, it has been verified that the LOO cross-validation algorithm based on ELM is efficient and has good generalization performance.

### 3. OS-ELM with LOO Weighted Strategy

As shown above, in the training process of the classic OS-ELM algorithm, all samples are equally treated. As long as a new sample is arriving, the network weight will be updated. This rigid weight updating mechanism lacks adjustment flexibility according to the actual situation. Moreover, it tends to increase the unnecessary computation.

To improve the generalization ability of OS-ELM while maintaining model's simplicity, this paper improves this rigid weight updating mechanism of traditional OS-ELM effectively via adopting dynamic weighted strategy. This strategy determines each sample's importance according to its LOO error estimation in online scenario. Consequently, a new OS-ELM based on online LOO cross-validation weight-setting strategy (LW-OSELM) is proposed.

*3.1. LOO Error Estimation of ELM.* As discussed in Section 2, the fast LOO error estimation of ELM proposed by Feng et al. [24] derived that the generalization error in  $i$ th LOO iteration can be expressed as follows:

$$r_i = t_i - f_i(\mathbf{x}_i) = \frac{t_i - \mathbf{Hx}_i \mathbf{H}^\dagger \mathbf{T}}{1 - (\mathbf{Hx}_i \mathbf{H}^\dagger)_i}. \quad (12)$$

Obviously, (12) works mainly on offline learning setting rather than online sequential scenario. The key reason is that  $\mathbf{H}^+$  cannot be updated in online stage. Considering the sequential reaching of the sample in online setting, (12) can be extended to online sequential scenario, and it provides a channel to calculate the LOO error of each sample. The key is calculating  $\mathbf{H}^+$  from (12) in online manner. However, the time complexity of matrix inversion is  $O(n^3)$ , where  $n$  is the number of samples. Therefore, the modeling time will be significantly increased along with the number of training patterns. To avoid complex calculation and make the established model simple, we adopt the idea of block matrix inversion, which transforms the complex calculation into linear operation, for decreasing the computation greatly.

As pointed out by many theoretical researches, LOO error is almost unbiased estimation of true generalization performance. Once a sample's LOO error is smaller, this sample's contribution in the decision model is greater. In order to highlight the samples' contribution and ensure the generalization of models, we set the corresponding weights of each sample according to the value of LOO error in the process of online. At the same time, to ensure the simplicity of the model, the oldest sample, which has the furthest distance from the current moment, is eliminated. Namely, the samples cost is zero. To avoid complex calculation and make the established model simple, we follow the idea of block matrix inversion [26], which transforms the complex calculation into linear operation, for decreasing the computation greatly in online learning stage. Thus, a new kind of extreme learning machine based on online leave-one-out cross-validation is put forward, as in the following sections.

**3.2. Initial Stage of Training.** Suppose there are  $N$  training samples  $(\mathbf{x}_{i+1}, t_{i+1}), \dots, (\mathbf{x}_{i+N}, t_{i+N})$ . The hidden layer output matrix is  $\mathbf{H}_i = [\mathbf{h}_{i+1}^T \cdots \mathbf{h}_{i+N}^T]^T$ , and the output vector is  $\mathbf{T}_i = [t_{i+1} \cdots t_{i+N}]^T$ , calculating the output weight vector:

$$\boldsymbol{\beta}_i = \mathbf{H}_i^+ \mathbf{T}_i, \quad (13)$$

where

$$\mathbf{H}_i^+ = \mathbf{H}_i^T (\mathbf{H}_i \mathbf{H}_i^T)^{-1}. \quad (14)$$

Let  $\mathbf{A}_i = \mathbf{H}_i \mathbf{H}_i^T$ ; (14) can be rewritten as  $\mathbf{H}_i^+ = \mathbf{H}_i^T \mathbf{A}_i^{-1}$ .

**3.3. Add New Sample.** Add the new arrived sample  $(\mathbf{x}_{i+N+1}, t_{i+N+1})$  into training set. The output vector becomes  $\mathbf{T}_{i+1} = [t_{i+1} \cdots t_{i+N} \ t_{i+N+1}]^T = [\mathbf{T}_i^T \ t_{i+N+1}]^T$ , and the hidden layer matrix becomes  $\mathbf{H}_{i+1} = [\mathbf{H}_i^T \ \mathbf{h}_{i+N+1}^T]^T$ . Then we have

$$\mathbf{H}_{i+1}^+ = \mathbf{H}_{i+1}^T (\mathbf{H}_{i+1} \mathbf{H}_{i+1}^T)^{-1}. \quad (15)$$

Let  $\mathbf{A}_{i+1} = \mathbf{H}_{i+1} \mathbf{H}_{i+1}^T$ ; then

$$\mathbf{H}_{i+1}^+ = \mathbf{H}_{i+1}^T \mathbf{A}_{i+1}^{-1}, \quad (16)$$

because

$$\begin{aligned} \mathbf{A}_{i+1} &= \mathbf{H}_{i+1} \mathbf{H}_{i+1}^T = [\mathbf{H}_i^T \ \mathbf{h}_{i+N+1}^T]^T [\mathbf{H}_i^T \ \mathbf{h}_{i+N+1}^T] \\ &= \begin{bmatrix} \mathbf{H}_i \mathbf{H}_i^T & \mathbf{H}_i \mathbf{h}_{i+N+1}^T \\ \mathbf{h}_{i+N+1} \mathbf{H}_i^T & \mathbf{h}_{i+N+1} \mathbf{h}_{i+N+1}^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_i & \mathbf{H}_i \mathbf{h}_{i+N+1}^T \\ \mathbf{h}_{i+N+1} \mathbf{H}_i^T & \mathbf{h}_{i+N+1} \mathbf{h}_{i+N+1}^T \end{bmatrix}. \end{aligned} \quad (17)$$

For (17), according to Block matrix inversion, we have

$$\mathbf{A}_{i+1}^{-1} = \begin{bmatrix} \mathbf{A}_i^{-1} \\ \mathbf{0}^T \end{bmatrix} + \frac{1}{B} \begin{bmatrix} \mathbf{A}_i^{-1} \mathbf{C} \\ -1 \end{bmatrix} [\mathbf{C}^T \mathbf{A}_i^{-1} \ -1], \quad (18)$$

where  $B = \mathbf{h}_{i+N+1} \mathbf{h}_{i+N+1}^T - \mathbf{C}^T \mathbf{A}_i^{-1} \mathbf{C}$ ,  $\mathbf{C} = \mathbf{H}_i \mathbf{h}_{i+N+1}^T$ . So,  $\mathbf{A}_{i+1}^{-1}$  can be calculated based on  $\mathbf{A}_i^{-1}$ , which reduces computational cost largely. Then we have  $\mathbf{H}_{i+1}^+$  by substituting (18) into (16).

**3.4. Calculate the LOO Error.** Let  $\mathbf{H}_{i+1}^+$  set up the online LOO model; then the LOO error in  $i$ th LOO iteration can be expressed as follows:

$$r_j = t_j - f_j(\mathbf{x}_j) = \frac{t_j - \mathbf{H} \mathbf{x}_j \mathbf{H}_{i+1}^+ \mathbf{T}}{1 - (\mathbf{H} \mathbf{x}_j \mathbf{H}_{i+1}^+)_j}, \quad (19)$$

$$j = i + 1, i + 2, \dots, i + N + 1.$$

Then we can obtain the corresponding LOO error,  $r_j$ , of each sample from (19). According to the value of  $r_j$ , where  $j = i + 1, i + 2, \dots, i + N + 1$ , we set the relevant weight  $w_j$  of each sample, where  $0.98 \leq w_j \leq 1$ . Note that the smaller  $r_j$  is, the bigger  $w_j$  is. To emphasize the newest sample and make the decision model simple, we reset the weight  $w_{i+N+1}$  of the newest sample  $(\mathbf{x}_{i+N+1}, t_{i+N+1})$  as  $w'_{i+N+1}$ . This paper defines  $w'_{i+N+1}$  as 1.02. And we set the weight  $w_{i+1}$  of the oldest sample  $(\mathbf{x}_{i+1}, t_{i+1})$  as zero; namely, we set its contribution to the model as zero.

**3.5. Weighted Training.** After adding the new sample  $(\mathbf{x}_{i+N+1}, t_{i+N+1})$ , we set the weight  $w_{i+1}$  of oldest sample  $(\mathbf{x}_{i+1}, t_{i+1})$  as zero, namely, excluding this sample. After excluding  $(\mathbf{x}_{i+1}, t_{i+1})$ , the output vector becomes  $\mathbf{T}_{i+2} = [t_{i+2} \cdots t_{i+N+1}]^T$ , and the hidden matrix becomes  $\mathbf{H}_{i+2} = [\mathbf{h}_{i+2}^T \cdots \mathbf{h}_{i+N+1}^T]^T$ . Then we have

$$\mathbf{H}_{i+2}^+ = \mathbf{H}_{i+2}^T (\mathbf{H}_{i+2} \mathbf{H}_{i+2}^T)^{-1}. \quad (20)$$

Let  $\mathbf{A}_{i+2} = \mathbf{H}_{i+2} \mathbf{H}_{i+2}^T$ ; then

$$\mathbf{H}_{i+2}^+ = \mathbf{H}_{i+2}^T \mathbf{A}_{i+2}^{-1}. \quad (21)$$

From (21),  $\mathbf{H}_{i+2}^+$  contains two parts:  $\mathbf{H}_{i+2}^T$  and  $\mathbf{A}_{i+2}^{-1}$ . Because the calculation of  $\mathbf{A}_{i+2}^{-1}$  involves matrix inversion, we only set weight on  $\mathbf{H}_{i+2}^T$  in order to avoid the huge

computational cost in calculating LOO error. Then the hidden matrix

$$\mathbf{H}_{i+2} = [\mathbf{h}_{i+2}; \mathbf{h}_{i+3}; \dots; \mathbf{h}_{i+N}; \mathbf{h}_{i+N+1}] \quad (22)$$

becomes

$$\mathbf{H}_{i+2} = \begin{bmatrix} \mathbf{h}_{i+2} \\ \mathbf{h}_{i+3} \\ \vdots \\ \mathbf{h}_{i+N} \\ \mathbf{h}_{i+N+1} \end{bmatrix} \cdot \begin{bmatrix} w_{i+2} \\ w_{i+3} \\ \vdots \\ w_{i+N} \\ w_{i+N+1} \end{bmatrix} = \begin{bmatrix} w_{i+2}\mathbf{h}_{i+2} \\ w_{i+3}\mathbf{h}_{i+3} \\ \vdots \\ w_{i+N}\mathbf{h}_{i+N} \\ w_{i+N+1}\mathbf{h}_{i+N+1} \end{bmatrix}. \quad (23)$$

And  $\mathbf{H}_{i+2}^T$  becomes

$$\begin{aligned} \mathbf{H}_{i+2}^T &= [w_{i+2}\mathbf{h}_{i+2}^T, w_{i+3}\mathbf{h}_{i+3}^T, \dots, w_{i+N}\mathbf{h}_{i+N}^T, w_{i+N+1}\mathbf{h}_{i+N+1}^T], \\ &\quad (24) \end{aligned}$$

where  $w_{i+2}, w_{i+3}, \dots, w_{i+N}, w_{i+N+1}$  are the corresponding weights  $0.98 \leq w_j \leq 1$ ,  $j = i + 2, i + 3, \dots, i + N$ , and  $w_{i+N+1} = w'_{i+N+1} = 1.02$ , because

$$\begin{aligned} \mathbf{A}_{i+1} &= \mathbf{H}_{i+1}\mathbf{H}_{i+1}^T = [\mathbf{h}_i^T \quad \mathbf{H}_{i+2}^T]^T [\mathbf{h}_i^T \quad \mathbf{H}_{i+2}^T] \\ &= \begin{bmatrix} \mathbf{h}_i\mathbf{h}_i^T & \mathbf{h}_i\mathbf{H}_{i+2}^T \\ \mathbf{H}_{i+2}\mathbf{h}_i^T & \mathbf{H}_{i+2}\mathbf{H}_{i+2}^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{h}_i\mathbf{h}_i^T & \mathbf{h}_i\mathbf{H}_{i+2}^T \\ \mathbf{H}_{i+2}\mathbf{h}_i^T & \mathbf{A}_{i+2} \end{bmatrix}. \end{aligned} \quad (25)$$

From (25), there is a relationship between  $\mathbf{A}_{i+2}$  and  $\mathbf{A}_{i+1}$ . So  $\mathbf{A}_{i+2}^{-1}$  can be calculated on the basis of  $\mathbf{A}_{i+1}^{-1}$  to simplify the calculation.

Assume that  $\mathbf{A}_{i+1}^{-1}$  can be partitioned and expressed as follows:

$$\mathbf{A}_{i+1}^{-1} = \begin{bmatrix} a & \mathbf{F}^T \\ \mathbf{F} & \mathbf{G} \end{bmatrix}, \quad (26)$$

where  $a \in \mathbf{R}$ ,  $\mathbf{F} \in \mathbf{R}^N$ , and  $\mathbf{G} \in \mathbf{R}^{N \times N}$ .

As in (25), let  $g = \mathbf{h}_i\mathbf{h}_i^T$ ,  $\mathbf{P} = \mathbf{h}_i\mathbf{H}_{i+2}^T$ ; then (25) is equivalent to

$$\mathbf{A}_{i+1} = \begin{bmatrix} g & \mathbf{P} \\ \mathbf{P}^T & \mathbf{A}_{i+2} \end{bmatrix}. \quad (27)$$

By the definition of matrix inversion,  $\mathbf{A}_{i+1}\mathbf{A}_{i+1}^{-1} = \mathbf{E}_{(N+1) \times (N+1)}$ ; namely,

$$\begin{bmatrix} g & \mathbf{P} \\ \mathbf{P}^T & \mathbf{A}_{i+2} \end{bmatrix} \begin{bmatrix} a & \mathbf{F}^T \\ \mathbf{F} & \mathbf{G} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_{N \times N} \end{bmatrix}. \quad (28)$$

Through the block matrix multiplication, we have

$$\begin{aligned} \mathbf{P}^T a + \mathbf{A}_{i+2}\mathbf{F} &= \mathbf{0}, \\ \mathbf{P}^T \mathbf{F}^T + \mathbf{A}_{i+2}\mathbf{G} &= \mathbf{E}_{N \times N}. \end{aligned} \quad (29)$$

Calculating (29), we have

$$\mathbf{A}_{i+2}^{-1} = \mathbf{G} - \frac{\mathbf{F}\mathbf{F}^T}{a}. \quad (30)$$

Thus, we have  $\mathbf{H}_{i+2}^+$  by substituting (24) and (30) into (21).

Then we can update the network weights according to the following equation:

$$\boldsymbol{\beta} = \mathbf{H}_{i+2}^+ \mathbf{T}_{i+2}. \quad (31)$$

**3.6. Algorithm.** The algorithm of LW-OSELM can be described into two steps.

*Step 1* (The initial stage of training). (1) For the initial training sample set  $D_0 = \{(\mathbf{x}_i, t_i) \mid i = 1, 2, \dots, N_0\}$ , calculate the network weights based on the  $N_0$  training samples by the following equation:

$$\begin{aligned} \boldsymbol{\beta}_0 &= \mathbf{H}_0^+ \mathbf{T}_0, \\ \mathbf{H}_0^+ &= \mathbf{H}_0^T \mathbf{A}_0^{-1}, \\ \mathbf{A}_0 &= \mathbf{H}_0 \mathbf{H}_0^T, \end{aligned} \quad (32)$$

where  $\mathbf{H}_0$  is the hidden layer matrix based on new training set and  $\mathbf{T}_0 = [t_1, t_2, \dots, t_{N_0}]$ .

(2) Set  $k = 0$ .

*Step 2* (Online learning stage). (1) Set the  $(k + 1)$ th arrived sample as  $(\mathbf{x}_{N_0+k+1}, t_{N_0+k+1})$ . Now the output vector becomes  $\mathbf{T}_{k+1} = [\mathbf{T}_k^T \quad t_{N_0+k+1}]^T$ , and the hidden layer matrix becomes  $\mathbf{H}_{k+1} = [\mathbf{H}_k^T \quad \mathbf{h}_{N_0+k+1}^T]^T$ . Calculate  $\mathbf{H}_{k+1}^+$ :

$$\mathbf{H}_{k+1}^+ = \mathbf{H}_{k+1}^T \mathbf{A}_{k+1}^{-1}, \quad (33)$$

where

$$\begin{aligned} \mathbf{A}_{k+1}^{-1} &= \begin{bmatrix} \mathbf{A}_k^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{B} \begin{bmatrix} \mathbf{A}_k^{-1} \mathbf{C} \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{C}^T \mathbf{A}_k^{-1} & -1 \end{bmatrix}, \\ \mathbf{B} &= \mathbf{h}_{N_0+k+1} \mathbf{h}_{N_0+k+1}^T - \mathbf{C}^T \mathbf{A}_k^{-1} \mathbf{C}, \end{aligned} \quad (34)$$

$$\mathbf{C} = \mathbf{H}_k \mathbf{h}_{N_0+k+1}^T.$$

(2) Calculate the LOO error of each sample by the following equation:

$$\begin{aligned} r_j &= t_j - f_j(\mathbf{x}_j) = \frac{t_j - \mathbf{H}\mathbf{x}_j \mathbf{H}_{k+1}^+ \mathbf{T}}{1 - (\mathbf{H}\mathbf{x}_j \mathbf{H}_{k+1}^+)_j}, \\ &\quad j = k + 1, k + 2, \dots, N_0 + k + 1. \end{aligned} \quad (35)$$

Then each sample is setting the corresponding weight  $w_j$  according to the value of  $r_j$ , where  $0.98 \leq w_j \leq 1$ , and the smaller the  $r_j$  is, the bigger the  $w_j$  is. In similar way, let  $w_{k+1} = 0$ ;  $w_{N_0+k+1} = w'_{i+N+1} = 1.02$ .

(3) Set the oldest samples weight  $w_{k+1} = 0$ . Now the output vector becomes  $\mathbf{T}_{k+2} = [t_{k+2} \cdots : t_{N_0+k+1}]^T$ , and the hidden layer matrix becomes

$$\begin{aligned} \mathbf{H}_{k+2} \\ = [w_{k+2}\mathbf{h}_{k+2}; w_{k+3}\mathbf{h}_{k+3}; \dots, w_{N_0+k}\mathbf{h}_{N_0+k}; w_{N_0+k+1}\mathbf{h}_{N_0+k+1}]. \end{aligned} \quad (36)$$

Calculate  $\mathbf{H}_{k+2}^+$  by the following equation:

$$\mathbf{H}_{k+2}^+ = \mathbf{H}_{k+2}^T \mathbf{A}_{k+2}^{-1}. \quad (37)$$

Make  $\mathbf{A}_{k+1}^{-1}$  represented as a new block:

$$\mathbf{A}_{k+1}^{-1} = \begin{bmatrix} a & \mathbf{F}^T \\ \mathbf{F} & \mathbf{G} \end{bmatrix}. \quad (38)$$

In (37), consider

$$\begin{aligned} \mathbf{A}_{k+2}^{-1} &= \mathbf{G} - \frac{\mathbf{F}\mathbf{F}^T}{a}, \\ \mathbf{H}_{k+2}^T \\ &= [w_{k+2}\mathbf{h}_{k+2}^T, w_{k+3}\mathbf{h}_{k+3}^T, \dots, w_{N_0+k}\mathbf{h}_{N_0+k}^T, \\ &\quad w_{N_0+k+1}\mathbf{h}_{N_0+k+1}^T]. \end{aligned} \quad (39)$$

Then substitute (39) into (37) to get  $\mathbf{H}_{k+2}^+$ .

(4) Use (40) to update the network weight:

$$\beta_{k+1} = \mathbf{H}_{k+2}^+ \mathbf{T}_{k+2}. \quad (40)$$

(5) Let  $\mathbf{H}_k = \mathbf{H}_{k+2}$ ,  $\mathbf{T}_k = \mathbf{T}_{k+2}$ , and  $\mathbf{A}_k^{-1} = \mathbf{A}_{k+2}^{-1}$ . Let  $k = k + 1$ , and go to step (2).

For better understanding, here we provide a flow chart of the proposed algorithm, shown as in Figure 1.

#### 4. Experimental Results

In this section, we run experiments to test the proposed algorithm. Our goal is to demonstrate that the proposed algorithm can efficiently improve the generalization performance of OS-ELM in time series prediction. For comparison, we choose two baselines: the classical ELM [9] and OS-ELM [12]. The proposed OS-ELM algorithm with fast LOO weighted strategy is named as LW-OSELM.

For completeness, we examine two types of nonstationary time series data sets. We start with two benchmark chaotic time series data. We further present results on a real-world data set, that is, air pollutants forecasting in Macau [27]. These data sets are described below. The goal is to test the generalization performance on time series data as well as the running speed. In each experiment, all results are the mean of 100 trials. RBF activation function is used in each algorithm. Each variable is linearly rescaled. A method with higher classification accuracy is better.

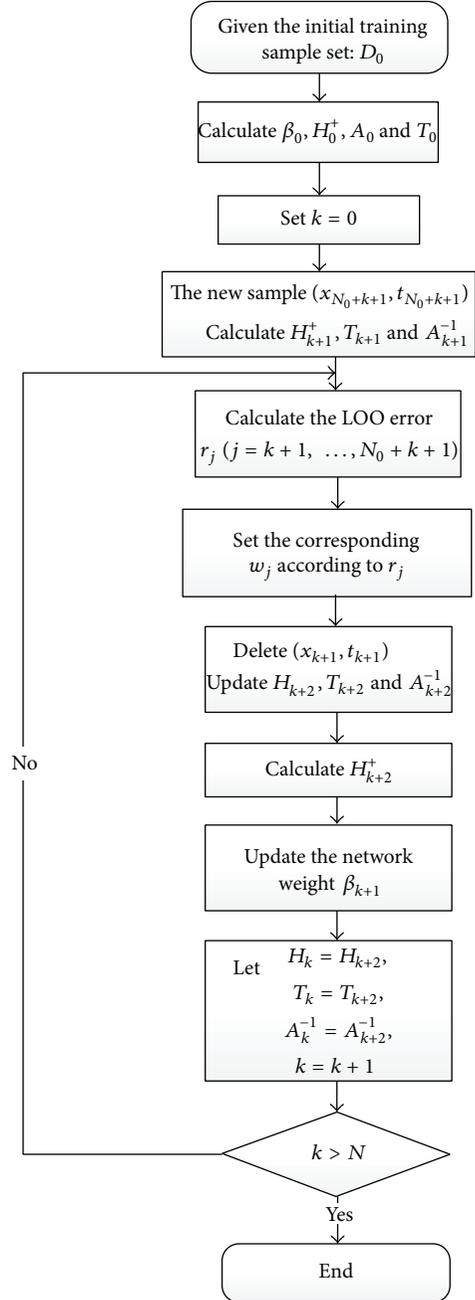


FIGURE 1: Flow chart of the proposed algorithm.

**4.1. Chaotic Time Series.** To verify the effectiveness of LW-OSELM for online time series prediction, we take the simulation experiment on three benchmark chaotic time series data: Mackey-Glass, Henon, and Lorenz.

As a chaotic map widely used, Mackey-Glass time series is shown as the following equation:

$$\frac{dx(t)}{dt} = \frac{ax(t - \text{TAU})}{1 + x^{10}(t - \text{TAU})} - bx(t). \quad (41)$$

The initial values are  $a = 0.2$ ,  $b = 0.1$ , and  $\text{TAU} = 30$ . Here we generate a set of 1300 points with the first 700 points for

training and the last 600 points for test.  $\tau$  is set as 1, and the embedding dimension  $m$  is set as 4.

Henon time series is shown as the following equation (42):

$$\begin{aligned} x(k+1) &= 1 - A(x(k))^2 + y(k), \\ y(k+1) &= Bx(k). \end{aligned} \quad (42)$$

The initial values are  $A = 1.4$ ,  $B = 0.3$ ,  $x(0) = 0.03$ , and  $y(0) = 0.02$ . We generate a set of 600 points with the first 500 points for training and last 100 points for test. In phase reconstruction stage,  $\tau$  is set as 1, and the embedding dimension  $m$  is set as 3.

Lorenz time series is shown as the following equation:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= rx - y - xz, \\ \frac{dz}{dt} &= -bz + xy. \end{aligned} \quad (43)$$

The initial values are  $\sigma = 16$ ,  $b = 4$ ,  $r = 45.92$ ,  $x(0) = 0.03$ , and  $y(0) = 0.02$ . We generate a set of 1600 points with the first 600 points for training and last 1000 points for test. In phase reconstruction stage,  $\tau$  is set as 1, and the embedding dimension  $m$  is set as 6.

First, we report the numerical results on the three standard data sets. The mean RMSE of 100 trials on the three data sets are listed in Tables 1, 2, and 3. Here the numbers of neurons are 16, 25, and 25, respectively.

As shown in Tables 1 to 3, although the training time of LW-OSELM is the longest, it gets not only the least training errors, but also the least test errors compared to the others, which is the key factor to measure the performance of a model. Specifically speaking, on Mackey-Glass data, LW-OSELM gets 40% improvement against ELM and 10.6% against OS-ELM in terms of test error. These two values are 4.8% and 4.4%, respectively. In addition, the training times of LW-OSELM on two data sets are 2.844 and 0.2194, respectively, which is in an acceptable range, so it does not affect the model's performance too much. Therefore, LW-OSELM has better performance than the ELM and OS-ELM.

As the only adjustable parameters, it is necessary to test the performance with the change of hidden neurons. Figures 2 and 3, respectively, illustrate the change of the training error and test error of LW-OSELM, OS-ELM, and ELM, with different number of hidden neurons.

From Figures 2 and 3, it is obvious that the change trends of training error and test error of three algorithms are roughly the same, which shows that LW-OSELM is meaningful, and LW-OSELM has smaller errors than the others with the same hidden neurons in most cases. So we can declare that the LW-OSELM has better performance.

In order to further prove the comparative results, we illustrate the predictive performance of three models on data set Mackey-Glass, Henon, and Lorenz, as shown in Figures 4, 5, and 6, respectively.

TABLE 1: Result of three algorithms on Mackey-Glass data set.

|                  | ELM    | OS-ELM | LW-OSELM |
|------------------|--------|--------|----------|
| Training time(s) | 0.0052 | 0.0832 | 2.8444   |
| Test time(s)     | 0.0009 | 0.0021 | 0.0156   |
| Training error   | 0.0072 | 0.0046 | 0.0041   |
| Test error       | 0.0070 | 0.0047 | 0.0042   |

TABLE 2: Result of three algorithms on Henon data set.

|                  | ELM    | OS-ELM | LW-OSELM |
|------------------|--------|--------|----------|
| Training time(s) | 0.0070 | 0.0507 | 0.0554   |
| Test time(s)     | 0.0062 | 0.0031 | 0.0023   |
| Training error   | 0.0040 | 0.0050 | 0.0033   |
| Test error       | 0.0042 | 0.0052 | 0.0034   |

TABLE 3: Result of three algorithms on Lorenz data set.

|                  | ELM    | OS-ELM | LW-OSELM |
|------------------|--------|--------|----------|
| Training time(s) | 0.1856 | 0.0585 | 5.6527   |
| Test Time(s)     | 0.0273 | 0.0250 | 0.0296   |
| Training error   | 0.0022 | 0.0012 | 0.0013   |
| Test error       | 0.0030 | 0.0020 | 0.0019   |

Note that the figures on the right column enlarge the predictive results for more illustration. The original data have no noise, so the predictive curve of three models are all very close to the real curve. From Figures 4(a) to 6(a), it is obvious that the predictive results of three models are roughly the same with real values, which proves the rationality of ELM-based methods. Furthermore, in enlarged part (we take it randomly) from Figures 4(b) to 6(b), we can easily find the prediction curve of LW-OSELM is nearer to the real curve than those of ELM and OS-ELM. These experimental results demonstrate that the proposed algorithm has higher generalization ability.

In brief, LW-OSELM has smaller training errors and test errors than ELM and OS-ELM. By calculating the LOO error of each sample, it is feasible to determine their importance. Through setting the corresponding weight for each sample by means of its LOO error estimation, LW-OSELM is endowed better generalization and stability.

**4.2. Macao Meteorological Time Series.** In application of air quality monitoring where data often reach in online sequential manner, it is a typical kind of nonstationary time series. In order to further test the stability and generalization of LW-OSELM, we choose suspended particulate matters (PM<sub>10</sub>) to conduct experiment. Due to the limitation of the acquisition data, this paper adopts the air quality data of Macao meteorological bureau to conduct simulation experiments [24].

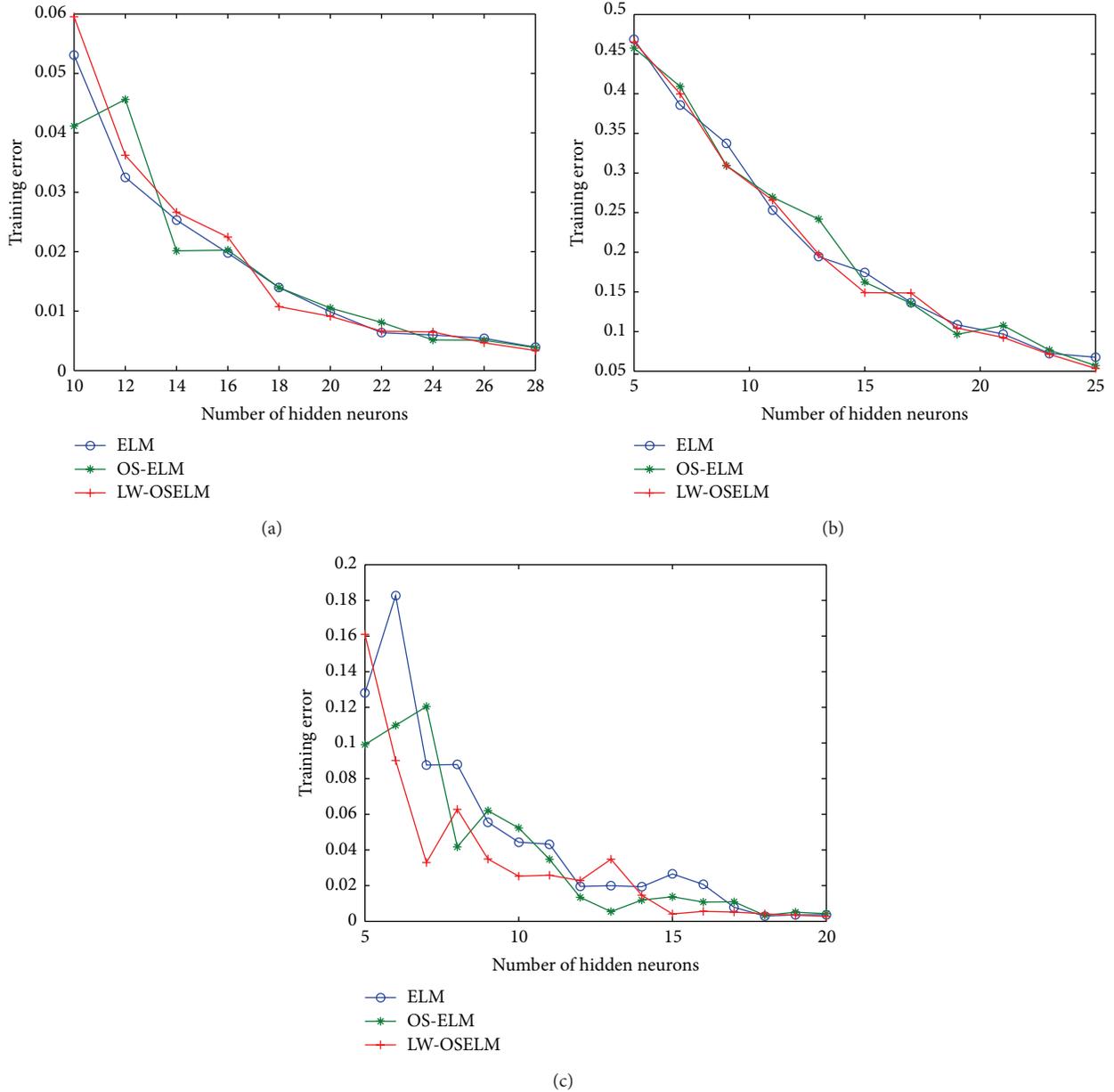


FIGURE 2: Training error of three algorithms with different number of hidden neurons for data sets (a) Mackey-Glass, (b) Henon, and (c) Lorenz.

**4.2.1. Data Pre-Processing.** We define the training data set  $D = (x, t)$ , where  $x$  represents the input variables and  $t$  represents the output variables. The input sample is constructed by the suspended particulate matters ( $PM_{10}$ ), nitrogen dioxide ( $NO_2$ ), and sulfur dioxide ( $SO_2$ ); that is,  $x = (d(PM_{10}), d(SO_2), d(NO_2), d(O_3))$ . And the output sample is the value of the next day's ( $PM_{10}$ ); that is,  $t = d + 1(PM_{10})$ . The training data and test data were normalized to a range of  $[-1, 1]$ , as follows:

$$v' = \frac{2(v - v_{\min})}{(v_{\max} - v_{\min})} - 1, \quad (44)$$

where  $v$  is a variable in  $x$  or  $t$ .  $v_{\max}$  and  $v_{\min}$  are the minimum value and maximum value of  $v$  in study period, respectively.

**4.2.2. Experiment Result.** To verify the validity of the LW-OSELM for time series prediction, we use the data collected from 2010 to 2013 to conduct experiment. Specifically, the data in 2010 are used for initial offline training, the data in 2011 are used for online training, the data in 2012 are used for test, and the data in 2013 are used for testing the generalization performance.

Considering the deficiency of traditional OS-ELM in time series prediction, we put forward a new kind of weighted extreme learning machine based on online leave-one-out

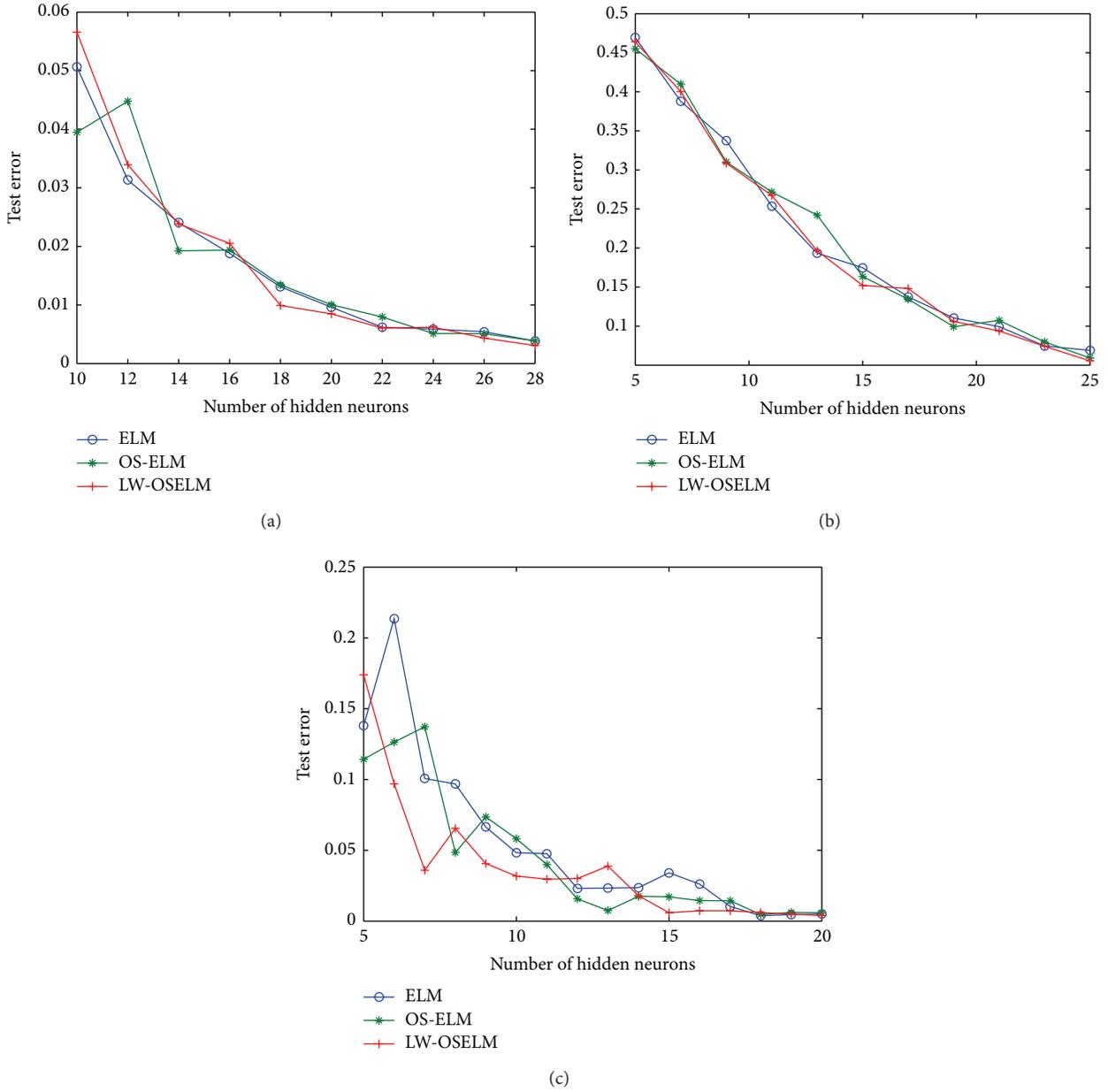


FIGURE 3: Test error of three algorithms with different number of hidden neurons for data sets (a) Mackey-Glass, (b) Henon, and (c) Lorenz.

cross-validation. According to the value of the online LOO error estimation, weigh the corresponding weight for each sample. To verify the rationality of this dynamic weight-setting strategy, it is necessary to compare LW-OSELM and the fixed-weighting OS-ELM (namely, WELM). The WELM will set weight value as 0.98 for all old samples and 1.02 for the latest sample. We set hidden neurons as 25. The comparison of LW-OSELM and WELM is illustrated in Table 4.

From Table 4, compared with WELM, the training error and test error of LW-OSELM are much smaller, which demonstrates that the dynamic weight-setting strategy, that is, setting weights on samples according to their online LOO errors, is feasible.

TABLE 4: Comparative results of LW-OSELM and WELM.

|                  | LW-OSELM | WELM   |
|------------------|----------|--------|
| Training time(s) | 2.9703   | 0.4056 |
| Test time(s)     | 0.0094   | 0      |
| Training error   | 0.1767   | 0.4276 |
| Test error       | 0.1794   | 0.4310 |

We compare the performance of ELM, OS-ELM, and LW-OSELM. Given the hidden layer activation function of RBF kernel function, the mean RMES of 100 trials on

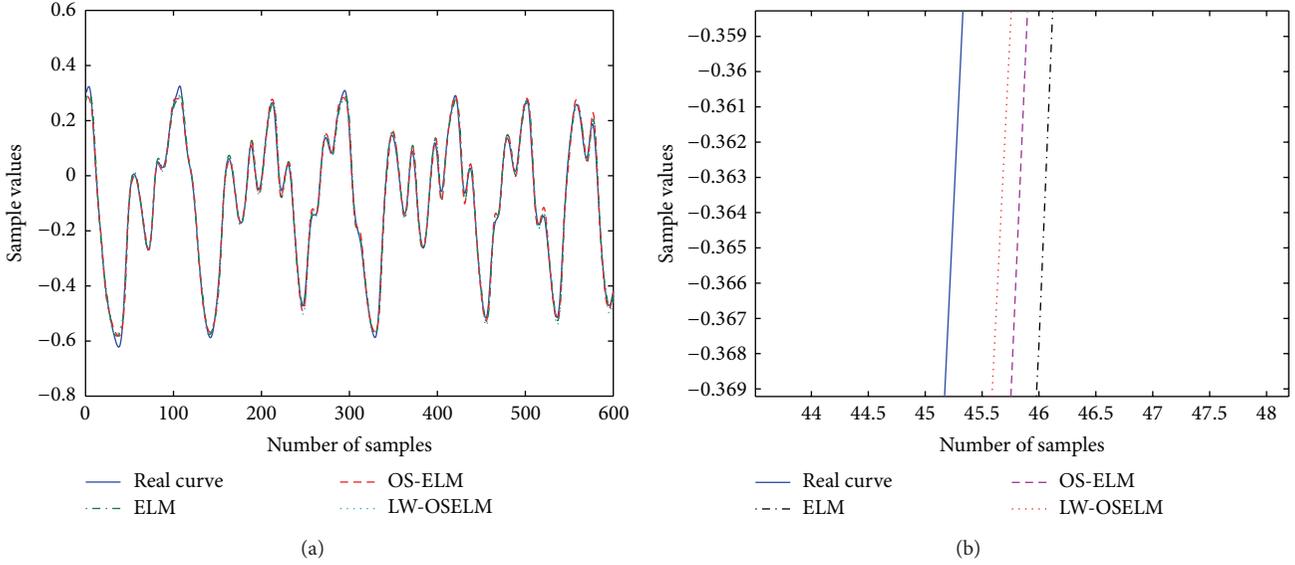


FIGURE 4: Comparative results of three models on data set Mackey-Glass (a) original figure and (b) enlarged figure.

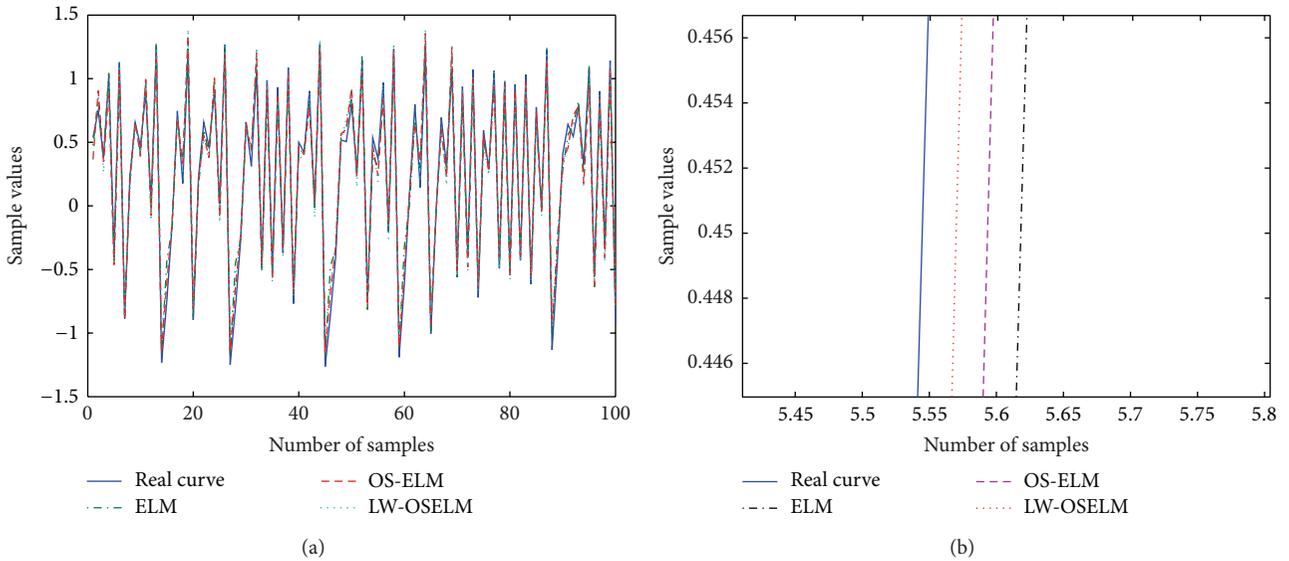


FIGURE 5: The comparison of real value and test values of three models for data set Henon (a) original figure and (b) enlarged figure.

Macao meteorological time series is listed in Table 5. Here the numbers of neurons are 15.

By contrast, LW-OSELM has little smaller training error and test error while its training time is a little longer. Note that, in this experiment, the comparative results are not remarkable like in Section 4.1. The reason is quite likely that we did not employ embedding dimension, that is, reconstructing phase space like in (1). Here we merely use the data in the latest day as input sample, rather than using the data of past few days.

We also examine the effect of hidden neurons. Figures 7 and 8, respectively, show the change of training error and test error with different number of hidden neurons.

TABLE 5: Comparative results of three models.

|                  | ELM    | OS-ELM | LW-OSELM |
|------------------|--------|--------|----------|
| Training time(s) | 0.0125 | 0.0998 | 3.3961   |
| Test time(s)     | 0.0031 | 0.0094 | 0.0012   |
| Training error   | 0.1706 | 0.1721 | 0.1705   |
| Test error       | 0.1811 | 0.1808 | 0.1739   |

From Figures 7 and 8, the training error and test error of LW-OSELM are both smaller than the others with the most hidden neurons. ELM tends to be the most unstable

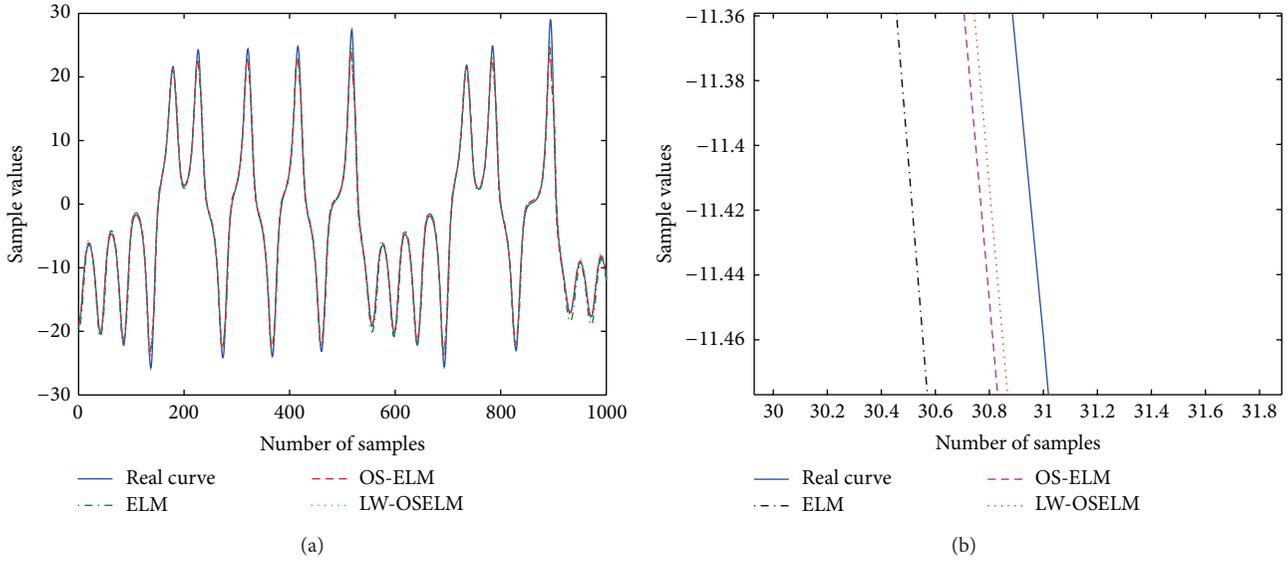


FIGURE 6: The comparison of real value and test values of three models for data set Lorenz (a) original figure and (b) enlarged figure.

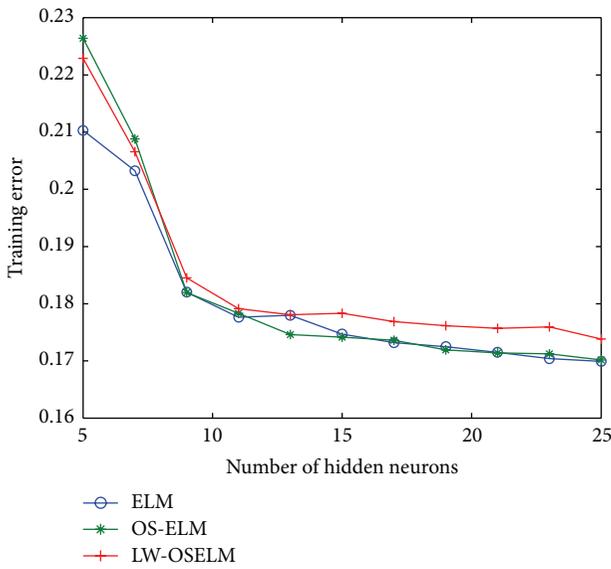


FIGURE 7: Training error of three models with different number of hidden neurons.

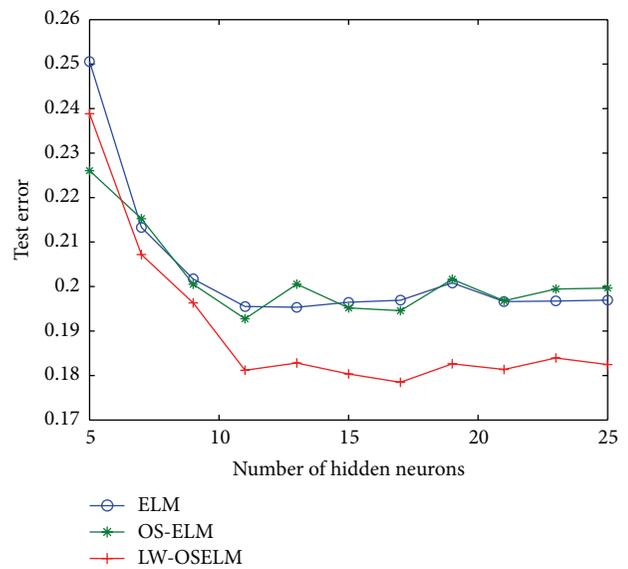


FIGURE 8: Test error of three models with different number of hidden neurons.

with drastic fluctuation. And LW-OSELM performs similarly stable to OS-ELM, which keeps pace with the results in Table 5.

Moreover, we report the generalization performance of three algorithms with different prediction step, as in Figure 9.

Obviously, LW-OSELM is better than the others. For further clarification, we also compare the mean and variance of three models. The mean of ELM, OS-ELM, and LW-OSELM is, respectively, 0.0726, 0.0617, and 0.0605, and the variance is, respectively,  $3.8340e-004$ ,  $3.5584e-004$ , and  $3.0809e-004$ . Obviously, the average error and variance of LW-OSELM are both the least, which indicates that LW-OSELM has better accuracy and stronger stability.

### 5. Conclusion and Future Work

In this paper, nonstationary time series prediction is addressed. The key idea is distinguishing the importance of samples in time series using its LOO cross-validation error. This idea is a new attempt for weight-setting strategy. To realize this strategy, this paper utilizes OS-ELM as baseline algorithm and proposes a new LOO error estimation which is fast and quite applied to time series prediction. Based on this estimation, this paper proposes a dynamic weight-setting algorithm for OS-ELM. The experimental results on two benchmark chaotic time series data sets and a real-world data set demonstrate the effectiveness of the proposed approach. It is

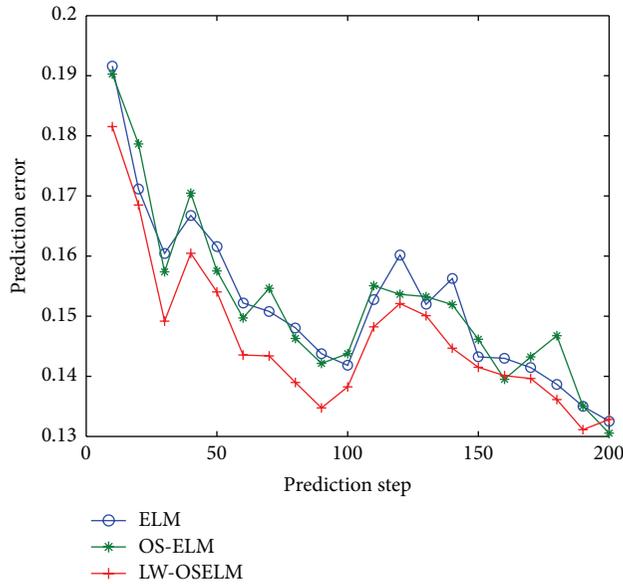


FIGURE 9: Accuracy of three algorithms with different prediction step.

a matter of choosing more efficient method for calculating leave-one-out error. Another problem is how to extend the proposed algorithm to multi-input multioutput regression, which should be achieved by introducing the corresponding background knowledge and will be studied in our future research.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors wish to thank the author C. M. Vong of [27] for useful discussion and instruction. This work was supported by the National Natural Science Foundation of China (nos. U1204609, 61173071), China Postdoctoral Science Foundation (no. 2014M550508), and Postgraduate Technology Innovation Project of Henan Normal University (no. YL201420).

## References

- [1] J. G. de Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.
- [2] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence*, Warwick, vol. 1981, pp. 366–381, 1980.
- [3] S. Mukherjee, E. Osuna, and F. Girosi, "Nonlinear prediction of chaotic time series using support vector machines," in *Proceedings of the 7th IEEE Workshop on Neural Networks for Signal Processing (NNSP '97)*, pp. 511–520, IEEE Press, Amelia Island, Fla, USA, September 1997.
- [4] K. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.
- [5] D. Du, X. Li, M. Fei, and G. W. Irwin, "A novel locally regularized automatic construction method for RBF neural models," *Neurocomputing*, vol. 98, pp. 4–11, 2012.
- [6] P. Yee and S. Haykin, "A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction," *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2503–2521, 1999.
- [7] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, 2014.
- [8] F. Corona and A. Lendasse, "Variable scaling for time series prediction," in *Proceedings of European Symposium on Time Series Prediction (ESTSP '07)*, pp. 69–76, Espoo, Finland, 2007.
- [9] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.
- [10] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [11] J. W. Cao, Z. Lin, G. B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [12] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [13] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.
- [14] F. E. H. Tay and L. J. Cao, "Modified support vector machines in financial time series forecasting," *Neurocomputing*, vol. 48, pp. 847–861, 2002.
- [15] W. Mao, G. Yan, and L. Dong, "Weighted solution path algorithm of support vector regression based on heuristic weight-setting optimization," *Neurocomputing*, vol. 73, no. 1-3, pp. 495–505, 2009.
- [16] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482–493, 2014.
- [17] X. Wang and M. Han, "Online sequential extreme learning machine with kernels for nonstationary time series prediction," *Neurocomputing*, vol. 145, pp. 90–97, 2014.
- [18] A. Grigorievskiy, Y. Miche, A.-M. Ventelä, E. Séverin, and A. Lendasse, "Long-term time series prediction using OP-ELM," *Neural Networks*, vol. 51, pp. 50–56, 2014.
- [19] H. Rong, G.-B. Huang, N. Sundarajan et al., "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.
- [20] D. M. Allen, "The relationship between variable selection and data augmentation and a method for prediction," *Technometrics*, vol. 16, pp. 125–127, 1974.
- [21] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization," *Neurocomputing*, vol. 74, no. 16, pp. 2413–2421, 2011.

- [22] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.
- [23] Y. Lan, Y. C. Soh, and G. B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16–18, pp. 3028–3038, 2010.
- [24] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352–1357, 2009.
- [25] X.-Y. Liu, P. Li, and C.-H. Gao, "Fast leave-one-out cross-validation algorithm for extreme learning machine," *Journal of Shanghai Jiaotong University*, vol. 45, no. 8, pp. 6–11, 2011.
- [26] X. Zhang and H.-L. Wang, "Local extreme learning machine and its application to condition on-line monitoring," *Journal of Shanghai Jiaotong University*, vol. 45, no. 2, pp. 236–240, 2011.
- [27] C.-M. Vong, W.-F. Ip, P.-K. Wong, and C.-C. Chiu, "Predicting minority class for suspended particulate matters level by extreme learning machine," *Neurocomputing*, vol. 128, pp. 136–144, 2014.

## Research Article

# Data-Driven Dynamic Modeling for Prediction of Molten Iron Silicon Content Using ELM with Self-Feedback

Ping Zhou,<sup>1</sup> Meng Yuan,<sup>1</sup> Hong Wang,<sup>1,2</sup> and Tianyou Chai<sup>1</sup>

<sup>1</sup>State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China

<sup>2</sup>Control System Center, Manchester University, Manchester M60 1QD, UK

Correspondence should be addressed to Ping Zhou; [zhouping@mail.neu.edu.cn](mailto:zhouping@mail.neu.edu.cn)

Received 21 August 2014; Revised 17 November 2014; Accepted 18 November 2014

Academic Editor: Jiuwen Cao

Copyright © 2015 Ping Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Silicon content ([Si] for short) of the molten metal is an important index reflecting the product quality and thermal status of the blast furnace (BF) ironmaking process. Since the online detection of [Si] is difficult and larger time delay exists in the offline assay procedure, quality modeling is required to achieve online estimation of [Si]. Focusing on this problem, a data-driven dynamic modeling method is proposed using improved extreme learning machine (ELM) with the help of principle component analysis (PCA). First, data-driven PCA is introduced to pick out the most pivotal variables from multitudinous factors to serve as the secondary variables of modeling. Second, a novel data-driven ELM modeling technology with good generalization performance and nonlinear mapping capability is presented by applying a self-feedback structure on traditional ELM. The feedback outputs at previous time together with input variables at different time constitute a dynamic ELM structure which has a storage ability to tackle data in different time and overcomes the limitation of static modeling of traditional ELM. At last, industrial experiments demonstrate that the proposed method has a better modeling and estimating accuracy as well as a faster learning speed when compared with different modeling methods with different model structures.

## 1. Introduction

Blast furnace (BF) is a giant countercurrent reactor and heat exchanger in metallurgical industry and is the first step towards the production of steel [1]. During the BF ironmaking system working, the solid raw materials including iron ore and coke are charged layer by layer from the top of the BF, while the compressed air and some auxiliary fuels are introduced through tuyeres just equipped above the hearth for smelting to produce molten iron. The complex chemical reactions and transport phenomena take place in the different zones along the top to the bottom of the BF. Gas-solid, solid-solid, and solid-liquid phases interacted in it and are accompanied with features like high temperature, high pressure, multiphase coupling, and multiphysics field which coexist simultaneously [1–3]. As one of the most complex industrial reactors, the BF has received broad interests both theoretically and experimentally due to its complexity and the key role of iron and steel industry on national economy.

However, it is true that the operation and control of an industrial BF is a serious problem and still relies on the manual operation of foremen experientially [1, 2]. So far, there remain some open problems both in metallurgical fields and in engineering control fields, such as the closed-loop control or operational optimization for the whole BF ironmaking process [4–6].

Undoubtedly, the most crucial obstacle for closed-loop control of BF is that the current regular instruments do not have the ability to feed the need of online measurement for molten iron quality, such as the silicon content ([Si]) in the final hot metal. In the past decades, through continuous efforts and attempts, a great number of models and algorithms have been developed trying to tackle the modeling problem for silicon content prediction. These existing methods include linear model based methods like ARX and ARMAX models [6–8], partial least squares based methods [9], and nonlinear intelligent based methods like artificial neural network (ANN) model [10–12] and support

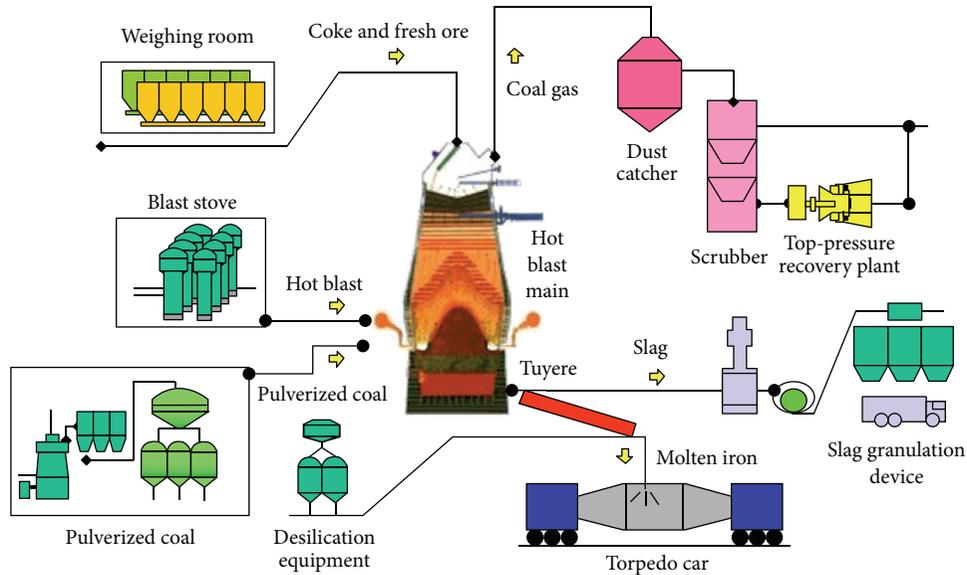


FIGURE 1: Schematic diagram of a typical BF ironmaking process.

vector machine (SVM) model [1, 2, 13, 14]. Though these existing methods have made some achievements in practical application, most of these studies are only focused on the static modeling for [Si] prediction while little attention has been paid to dynamical modeling of this quality parameter.

The BF ironmaking process is a complicated dynamic system with many influential factors and large time lag. To capture the system dynamics, the time series and time delays of the relevant input and output variables should be taken into account during the process modeling. This also means that the existing static prediction models cannot capture the process nonlinear dynamics very well and thus do not provide much accuracy estimation. Therefore, the self-feedback structure which can construct a dynamic system may appear more important for the BF system with serious nonlinear dynamics and large time lag. Moreover, most of the existing prediction models are trained by gradient-based algorithms such as back propagation (BP) algorithm and its variants. It is clear that the learning speed of such intelligent models is insufficiently fast as larger number of training data may be required. Moreover, the BP-like algorithm usually suffers from high computational burden, poor generalization ability, and local optima and overweighting problems [15].

On the other hand, a new machine learning approach that is termed as the extreme learning machine (ELM) has been recently proposed by Huang et al. in [15–18] and verified on a number of benchmark and real-world problems including pattern classification and prediction modeling [16–25]. The ELM and its variants have been considered as a promising learning algorithm in contrast with other algorithms such as BP NN and SVM. This is because ELM has the following advantages: (1) much faster learning speed; (2) higher generalization performance in comparison with BP NN and SVM; and (3) no extra parameters needing to be tuned except the predefined network architecture [15–18, 22–27].

Based on the work of ELM proposed by Huang et al. [15–18], this paper proposed a data-driven dynamic modeling method to predict molten iron silicon content using ELM with the help of principle component analysis (PCA) [28, 29]. In the design procedure of this predictive model, data-driven PCA for reducing the input variables space of ELM has been constructed. Moreover, output self-feedback architecture has been introduced to establish a dynamic ELM model for practical BF dynamic system. This self-feedback structure enables ELM to overcome the static mapping limitation of its feedforward network structure. This improvement can further optimize the application of ELM in the area of dynamic time-series prediction. Lastly, performance of the proposed dynamic ELM based prediction model is compared with other well-known modeling algorithms by industrial experiments on 2<sup>#</sup> BF in Liuzhou Iron & Steel Group Co. of China.

## 2. Description of BF Ironmaking System

**2.1. Process Description.** The BF ironmaking is a continuous production process conducted in a closed vertical furnace where materials reduction from iron ore to molten iron takes place every time using carbon coke and gas in high temperature and high pressure environment. Due to the advantages like simple technology, high productivity, and high production efficiency, at present and a long period in the future, the BF smelting will still be the most important way of ironmaking. Indeed, due to the large quantity production, even small improvements of the process can result in considerable profit. Thus the ironmaking BF is regarded as a significant item in the economic development of any country.

Figure 1 is the schematic diagram of a typical BF ironmaking process, which mainly consists of hot blast main, feeding system, air supply system, gas filtration system, slag

treatment system, fuel injection system, and so forth. The inner part of a furnace main is divided into five zones: the throat, the stack, the belly, the bosh, and the hearth from top to bottom. When a BF ironmaking system runs, the solid raw materials consisting of coke and fresh ore are charged layer by layer with definite quantities from the top, while the preheated compressed air, together with pulverized coal, is introduced at the bottom through tuyeres, entering just above the hearth, which is a crucial region of BF where the final molten metal product gathers. The hot air at approximately 1200°C passes upward through the charge and reacts with the descending coke and the supplementary injected oil to generate carbon dioxide, which then changes to CO and H<sub>2</sub> at high temperature. A lot of heat energy is released during this period that can heat up the hearth as high as 2000°C. The generated CO and H<sub>2</sub> further reduce the descending iron ore to form hot metal accumulating in the hearth, and some unreduced impurities (mainly SiO<sub>2</sub>) form the slag (mainly CaSiO<sub>3</sub>) floating on the hot metal being lighter. The liquid hot metal and slag are periodically tapped out by opening clay-lined tapholes for the subsequent processing. Generally, it will take 6~8 hours for each period of BF ironmaking [30].

### 2.2. Importance of Modeling for Silicon Content Prediction.

For many countries, such as China, the steel industry is playing an important role in the national economy, and there are thus extensive interests in operational control and optimization of ironmaking BF for saving energy and reducing cost. Generally speaking, control of the BF system often means controlling the hot metal temperature and components, such as silicon content, sulfur content, and phosphorus content in hot metal within acceptable bounds, among which the silicon content is the most important one [31].

For a practical BF production process, silicon content ([Si]) is an important index indicating the chemical heat of molten iron. High silicon content means a large quantity of slag, and this would be easier to wipe off the phosphorus and sulphur in the hot metal. However, excessive silicon content will make cast iron stiff and brittle and even lead to lower yield of metal and easier splashing. In addition, high silicon content will result in a corresponding increase of SiO<sub>2</sub> in the slag, thereby influencing slagging speed of calclime, extending converting time, and intensifying corrosion to furnace lining. From an energy point of view, it would be desired to operate the BF process at low molten metal silicon content, still avoiding the risk of cooling the hearth which may result in chilled hearth. Generally, the content of silicon content should be controlled in 0.5%~0.7%.

Nowadays, it is still an insoluble dilemma to realize the closed-loop control of molten iron quality in ironmaking BF. The main bottleneck is that the direct online measurement on this quality parameter of molten iron is difficult to be realized with the existing conventional measuring means. Moreover, the offline assaying process for this index takes a long lag time, usually more than 1 hour. Therefore, online prediction based molten iron quality modeling must be established. Effective online prediction or estimation for silicon content not only

can offer useful information for operators to judge the inner smelting state and operational condition, but also plays a key role in realizing closed-loop control and operational optimization as well as energy-saving and cost-reducing.

## 3. Modeling Strategy Using PCA and ELM with Self-Feedback

Data-driven black-box model is a kind of input-output mode. It relies on the development of novel nonlinear signal processing and data analysis technologies along with computer hardware and software technologies and does not require any prior information about the process. The main thought of data-driven model is to approximate the input-output relationships using the strong nonlinear approximation power of some mathematical tools or artificial intelligence technologies, like artificial neural network, fuzzy logic, and support vector machines [32].

The proposed data-driven modeling strategy for silicon content prediction is shown in Figure 2. First, since the BF is a complicated high-dimensional nonlinear dynamic system combined with numerous coupled factors, data-driven PCA technology with a strong ability to handle high-dimensional nonlinear correlated data is introduced to pick a few key factors as the input variables of model so as to reduce the dimension and difficulty for prediction modeling. Then, considering that the BF ironmaking system is a nonlinear system with dynamic time-vary characteristic, the ELM with better nonlinear mapping and fast process capability modeling technology is brought in this paper. In the meantime, output self-feedback structure is put into use on the basis of traditional ELM in this method, and the output variables derived from previous time are fed back to the network input layer. These feedback outputs together with input variables at different time constitute a dynamic ELM structure which has a storage capacity and has the ability to tackle data in different time, thus overcoming the limitation of static modeling of traditional ELM.

*Remark 1.* As shown in Figure 2, the dynamic ELM based estimation model is developed to achieve the following nonlinear dynamic mapping:

$$y(t) = f_{\text{ELM}} \{X(t), \dots, X(t - k_1), y(t - 1), \dots, y(t - k_0)\}, \quad (1)$$

where  $X = [x_1, x_2, \dots, x_n]$  are the values of secondary variables selected by PCA and  $y$  is the quality parameter that needs to be estimated. The values of  $k_1, k_0 \in \mathbb{Z}^+$  are selected according to the time delays and time series of the relevant input and output variables and the sampling frequency of quality parameter  $y$  which is generally sampled slower than the process data  $X$  significantly.

*Remark 2.* Note that, in the learning period of the proposed ELM based dynamic estimation model using the training databases,  $y(t - 1), \dots, y(t - k_0)$  are the actual (sampling) values of  $y$ . After the proposed ELM model is trained and validated well, it will be applied in practice. Since the quality

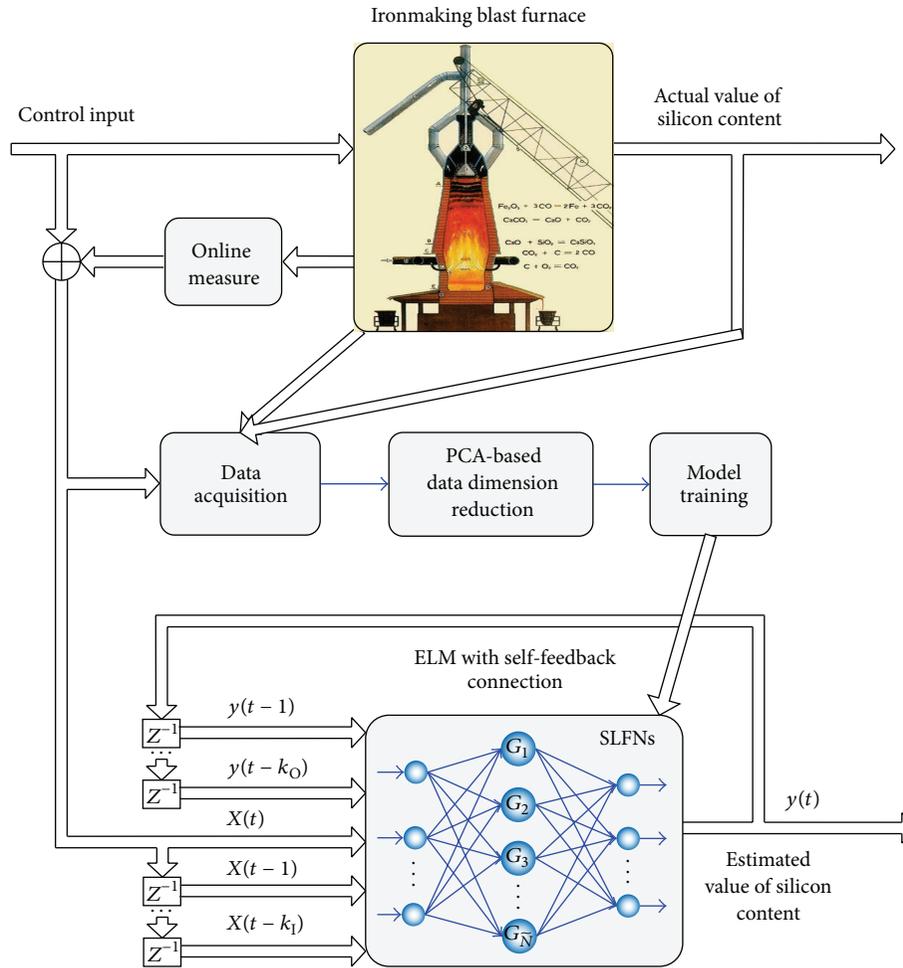


FIGURE 2: Strategy diagram of nonlinear intelligent modeling for silicon content prediction.

parameter  $y$  cannot be measured online and the offline assaying process takes a long time which is usually more than one hour, the actual value of  $y$  cannot be obtained in real time in practice. Therefore, to achieve the desired dynamic estimation of  $y$ , the estimated  $\hat{y}(t-1), \dots, \hat{y}(t-k_0)$  at past will be used to construct the self-feedback structure for the proposed dynamic ELM based estimation model.

*Remark 3.* The proposed modeling strategy has two advantages.

- (i) The dynamic property of time series and time delays is considered by introducing the output and inputs in previous time through a self-feedback structure. This self-feedback connection enables ELM to overcome the static mapping limitation of its feedforward network structure. Thus the improved version of ELM can capture the process nonlinear dynamics very well by remembering prior input and output states and using both the prior and current states to calculate new output value.

- (ii) Different from the BP-like modeling algorithm usually suffering from high computational burden, poor generalization ability, and local optima and overweighing problems, the ELM based modeling benefits from much faster learning speed, higher generalization performance, and ease of implantation and use (no extra parameters need to be tuned except the predefined network architecture).

## 4. Modeling Algorithm

*4.1. Selection of Secondary Variables by PCA-Based Dimension Reduction.* PCA is a kind of method trying to grasp the main contradiction part in statistical analysis process and analyze the main influencing factors from multiple objects in order to simplify the complex problems. Actually, the principle components conducted by PCA are the combination of column vectors picked by varimax from input matrix. Since correlations and noises always existed in practical industrial data, principle components with a small variance are usually some noisy information. Abandoning this data will not cause

a crucial information loss and can even achieve denoising to some extent.

Data set as shown in the following equation is considered here:

$$u_i = \mathbf{X}v_i, \quad (2)$$

where  $\mathbf{X}_{n \times m} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]$  is the measured  $n$  data array on  $m$  variables,  $u_i$  is the score vector, and  $v_i$  is the characteristic unit vector of covariance matrix  $\mathbf{X}^T \mathbf{X}$ , named load vector. The variance of  $u_i$  is  $\lambda_i$  which is also the eigenvalue of  $\mathbf{X}^T \mathbf{X}$  and satisfies  $\text{Var}(t_i) = \lambda_i, \lambda_1 \geq \dots \geq \lambda_m \geq 0$ . PCA is also a procedure used to explain the variance in a single data matrix. The principal component decomposition of  $\mathbf{X}$  in (2) can be represented as follows:

$$\mathbf{X} = \mathbf{U}\mathbf{V}^T = \sum_{i=1}^m u_i v_i^T + \mathbf{E}, \quad (3)$$

where  $u_i v_i^T$  is the  $i$ th principal component and  $\mathbf{E}$  is a matrix of residuals. It is to be noted that the score vectors are orthogonal and so are the loading vectors which are of unit length.

Equation (3) indicates that a rank  $n$  matrix  $\mathbf{X}$  can be decomposed as the sum of  $n$  rank 1 principal components. The number of principle components kept in (3) is determined by the total variance. The variance contribution and the total variance of principal component can be represented as follows:

$$\eta_k = \frac{\lambda_k}{\left(\sum_{j=1}^m \lambda_j\right)}, \quad (4)$$

$$C\eta_k = \sum_{i=1}^k \eta_i = \frac{\sum_{i=1}^k \lambda_i}{\sum_{j=1}^m \lambda_j},$$

where  $\eta_k$  is the  $k$ th principle component variance contribution and  $C\eta_k$  is the total variance of the first  $k$  terms. Usually, the total variance varies should be larger than 85%. Only in this case can the data dimension be reduced on the premise of not losing useful information.

After data dimension reduction and noise filtering through PCA, the data measurements are represented as

$$\mathbf{X} \approx \mathbf{U}_k \mathbf{V}_k^T = \sum_{i=1}^k u_i v_i^T, \quad (5)$$

where  $k$  is the number of remaining principle components,  $\mathbf{U}_k$  is the score vector of the first  $k$  terms, and  $\mathbf{V}_k$  is the loading vector of the first  $k$  terms.

*Remark 4.* A problem of the PCA-based dimension reduction is that the conducted principle components are comprehensive representation of the original higher-dimension physical variables. However, by computing the *component matrix* which contains the correlations between the principle component and the original physical variable, one can obtain the lower-dimension physical variables which related to the principle components mostly, according to some specific requirements.

*4.2. ELM with Self-Feedback Connection.* Extreme learning machine (ELM) is an algorithm for single hidden layer feedforward networks (SLFNs) with additive or radial basis function (RBF) hidden nodes whose learning speed can be thousands of times faster than conventional feedforward network learning algorithm like BP algorithm while reaching better approximation performance. In real application, net tends to be used for a finite data set. Huang and Babri prove that a SLFN with at most  $N$  hidden nodes and with almost any nonlinear activation function can learn  $N$  distinct observations with zero error [18]. And based on the work of [17], the SLFNs (with  $N$  hidden neurons) with arbitrary chosen input weights and bias were proved to have the ability to learn  $N$  distinct observations with arbitrary small error.

The procedure of the algorithm used here can be summarized as follows: for  $N$  arbitrary distinct samples  $(\mathbf{X}_i, \mathbf{Y}_i)$ , where  $\mathbf{X}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbf{R}^n$  and  $\mathbf{Y}_i = [y_{i1}, y_{i2}, \dots, y_{im}]^T \in \mathbf{R}^m$ , the output of a SLFN with  $\tilde{N}$  hidden nodes can be represented by

$$f_{\tilde{N}}(\mathbf{X}) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{a}_i, b_i, \mathbf{X}), \quad \mathbf{X} \in \mathbf{R}^n, \mathbf{a}_i \in \mathbf{R}^n, \quad (6)$$

where  $\mathbf{a}_i$  and  $b_i$  are the learning parameters of hidden nodes,  $\beta_i$  is the output weight, and  $G(\mathbf{a}_i, b_i, \mathbf{X})$  is the output of the  $i$ th hidden node with respect to the input data  $\mathbf{X}$ .

- (i) For additive hidden node,  $G(\mathbf{a}_i, b_i, \mathbf{X})$  is given by  $G(\mathbf{a}_i, b_i, \mathbf{X}) = g(\mathbf{a}_i \odot \mathbf{X} + b_i)$ ,  $b_i \in R$ , where  $\mathbf{a}_i$  is the input weight vector connecting the input layer to the  $i$ th hidden node,  $b_i$  is the bias of the  $i$ th hidden node, and  $\mathbf{a}_i \odot \mathbf{X}$  denotes the inner product of vector  $\mathbf{a}_i$  and  $\mathbf{X}$  in  $\mathbf{R}^n$ .
- (ii) For RBF hidden node,  $G(\mathbf{a}_i, b_i, \mathbf{X})$  is given by  $G(\mathbf{a}_i, b_i, \mathbf{X}) = g(b_i \|\mathbf{X} - \mathbf{a}_i\|)$ ,  $b_i \in R^+$ , where  $\mathbf{a}_i$  and  $b_i$  are the center and impact factor of  $i$ th RBF node and  $R^+$  indicates the set of all positive real values.

*Remark 5.* For the prediction modeling problem considered in this paper,  $\mathbf{X}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$  are the data of the model inputs variables selected by PCA.  $\mathbf{Y}_i = [y_{i1}, y_{i2}, \dots, y_{im}]^T$  are the data of molten iron silicon content that are to be estimated online, which means that  $m = 1$ . Moreover, the additive hidden node is used in our prediction modeling due to its simple structure.

In supervised batch learning, the learning algorithms use a finite number of input-output samples for training. For  $N$  arbitrary distinct samples  $(\mathbf{X}_i, \mathbf{Y}_i)$ , if a SLFN with  $\tilde{N}$  additive hidden nodes can approximate these  $N$  samples with zero error, it then implies that there exist  $\mathbf{a}_i, b_i$ , and  $\beta_i$  such that

$$f_{\tilde{N}}(\mathbf{X}_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(\mathbf{a}_i, b_i, \mathbf{X}) = \mathbf{Y}_j, \quad j = 1, \dots, N. \quad (7)$$

Equation (7) can be written compactly as

$$\mathbf{H}\beta = \mathbf{Y}, \quad (8)$$

where

$$\mathbf{H}(\mathbf{a}_1, \dots, \mathbf{a}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}, \mathbf{x}_1, \dots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{a}_1 \odot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{a}_{\tilde{N}} \odot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\mathbf{a}_1 \odot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{a}_{\tilde{N}} \odot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \quad (9)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}, \quad \mathbf{Y} = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times m}. \quad (10)$$

The purpose of ELM is training the net to find a least-squares solution  $\hat{\beta}$  of the linear system  $\mathbf{H}\beta = \mathbf{Y}$ :

$$\begin{aligned} & \|\mathbf{H}(a_1, \dots, a_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}) \hat{\beta} - \mathbf{Y}\| \\ & = \min_{\beta} \|\mathbf{H}(a_1, \dots, a_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}) \beta - \mathbf{Y}\|. \end{aligned} \quad (11)$$

And the solution of the above linear system can be solved by the inverse of matrix  $\beta$  by the Moore-Penrose method, which is

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{Y}, \quad (12)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of the hidden layer output matrix  $\mathbf{H}$  [15–17].

*Remark 6.* For the simplicity of the paper, the prediction modeling process based on ELM with additive hidden node is summarized as follows: giving a training set  $\mathbf{Z} = \{(\mathbf{X}_i, \mathbf{Y}_i) \mid \mathbf{X}_i \in \mathbf{R}^n, \mathbf{Y}_i \in \mathbf{R}^m, i = 1, \dots\}$  for prediction modeling and hidden neuron number  $\tilde{N}$ , the input weight  $\mathbf{a}_i$  and bias  $b_i$  can be assigned arbitrarily to calculate the output matrix  $\mathbf{H}$  of hidden layer by using (9). After that, the output weight  $\beta$  can be calculated by (12), which is essential for estimating output only based on estimating inputs.

*Remark 7.* The hidden node number  $\tilde{N}$  is the only parameter that needs to be predefined in the presented modeling method. In order to achieve optimal approximation ability of training and realize fast convergence aiming at complex industrial data, a proper (maybe optimal)  $\tilde{N}$  can be determined as the one which results in the lowest validation error through several trainings and validations.

## 5. Industrial Experiments

*5.1. Model Development.* In this section, a medium-sized blast furnace (as shown in Figure 3) with the working volume of 2000 m<sup>3</sup> in Liuzhou Iron & Steel Group Co. is chosen to perform the validation of the proposed silicon content prediction method. On the foundation of process mechanism and existing monitoring instruments status, measurable parameters influencing silicon content are determined as blast temperature (°C), blast pressure (kPa), oxygen enrichment percentage (%), flow rate of rich oxygen (m<sup>3</sup>/h), gas permeability (m<sup>3</sup>/min·kPa), gas volume of bosh (m<sup>3</sup>/min),



FIGURE 3: The 2<sup>#</sup> BF of Liuzhou Iron & Steel Group Co.

TABLE 1: Direct detecting parameters and their instrumentations.

| Variable (unit)                              | Notation | Instrumentation (notation)                         |
|----------------------------------------------|----------|----------------------------------------------------|
| Flow rate of cold air (m <sup>3</sup> /min)  | $q_c$    | HH-WLB differential pressure flowmeter (FT)        |
| Flow rate of rich oxygen (m <sup>3</sup> /h) | $q_o$    | A+K balance flowmeter (FT)                         |
| Blast pressure (kPa)                         | $p_h$    | DPharp EJA high accuracy pressure transmitter (PT) |
| Furnace top pressure (kPa)                   | $p_f$    | DPharp EJA high accuracy pressure transmitter (PT) |
| Blast temperature (°C)                       | $t_h$    | Hongguang SBW temperature transmitter (TT)         |
| Blast humidity (g/m <sup>3</sup> )           | $h_c$    | Air humidity sensor (HT)                           |

bosh gas index (m<sup>3</sup>/min·m<sup>2</sup>), blast kinetic energy (kJ/s), blast humidity (g/m<sup>3</sup>), flow rate of cold air (m<sup>3</sup>/h), feed blast ratio (wt%), resistance coefficient, volume of coal injection (kg/t), theoretical burning temperature (°C), actual wind speed (m/s), and furnace top pressure (kPa). Figure 4 is a schematic diagram of this actual BF ironmaking system and its measurement system. Through this figure, one can get an intuitive understanding of the detection position distribution of each measurable parameter. In Figure 4, the direct detecting parameters are explained as shown in Table 1, and the indirect detecting variables and their calculation formulas by the direct detecting parameters are listed in Table 2.

Considering the impact of strong correlation between the selected 16 input variables, PCA is used to determine the key input variables that influence the molten iron silicon content mostly. According to (4), the eigenvalue and the variance contribution rate of each component can be calculated as shown in Figure 5. It can be summarized that the cumulative variance contribution rate of the first 6 terms is 98.723%. This means these 6 principal components are sufficient to describe the major variances in the data. Then, by computing the component matrix of principle components, 6 process variables can be determined as the secondary input of the [Si] prediction model. These secondary variables include hot blast pressure  $x_1$  (kPa), hot blast temperature  $x_2$  (°C), oxygen enrichment percentage  $x_3$  (%), volume of coal injection  $x_4$

TABLE 2: Indirect detecting parameters and their calculation formulas.

| Variable (unit)                                 | Calculation formulas                                                                                                      |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| Oxygen enrichment percentage (%)                | $((0.0163q_o + ((0.21 + (0.29h_c/800)) \times (q_c/60))) / ((q_c/60) + (q_o/60)) - (0.21 + (0.29h_c/800))) \times 100$    |
| Gas permeability ( $m^3/min \cdot kPa$ )        | $100q_c / (p_h - p_f)$                                                                                                    |
| Gas volume of bosh (GVB) ( $m^3/min$ )          | $(1.21q_c/60) + (q_o/30) + (44.8h_cq_c/6000) + (44.8h_cq_o/6000) + ((22.4VCI \times \text{Hydrogen content in coal})/12)$ |
| Bosh gas index ( $m^3/(min \cdot m^2)$ )        | $GVB/78.5398125$                                                                                                          |
| Blast kinetic energy (kJ/s)                     | $(0.021q_c + ((q_c h_c/60000) + (q_o h_c/60000)) / (1 - (h_c/803.6))) / 0.2AWS^2/50$                                      |
| Feed blast ratio (wt%)                          | $q_c/2000$                                                                                                                |
| Resistance coefficient                          | $((10000p_h^2 - 100p_f^2) / GVB)^{1.7}$                                                                                   |
| Theoretical burning temperature ( $^{\circ}C$ ) | $1559 + (0.839t_h) + (4972q_o/q_c) - (6.033h_c) - (3.15VCI \times 1000000/q_c)$                                           |
| Actual wind speed (AWS) (m/s)                   | $0.101325(273 + t_h) / (273(0.101325 + p_h)) \times (q_c/3600 \times 4/3.14/30)$                                          |

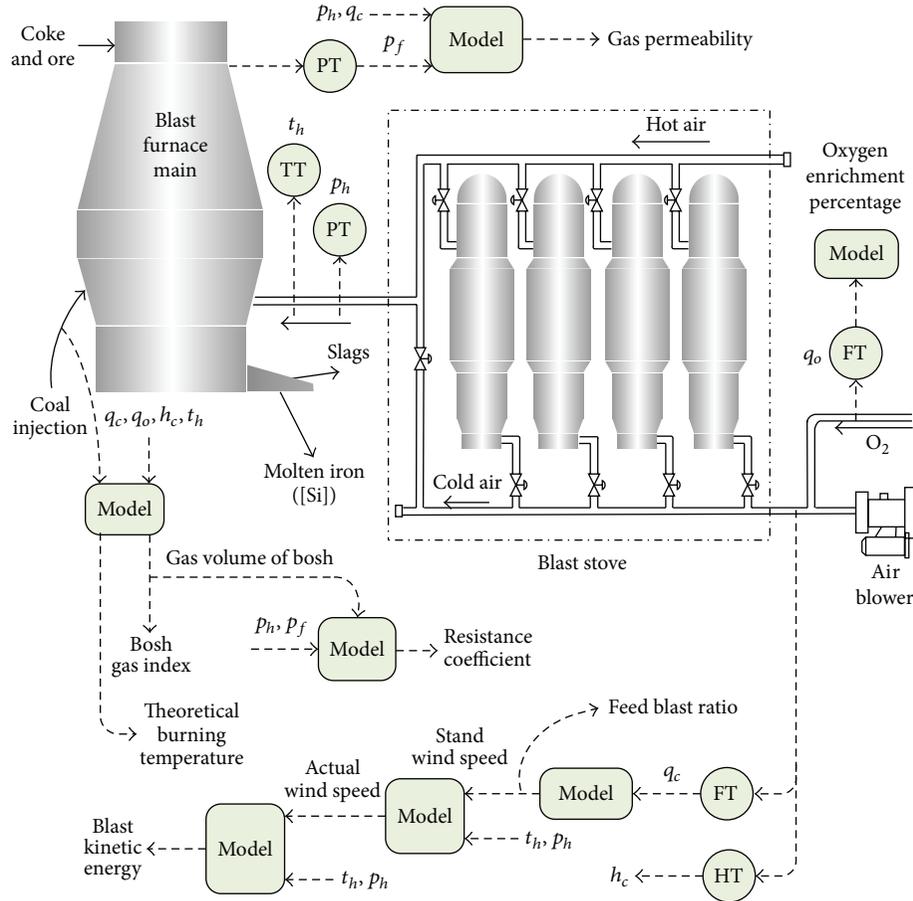


FIGURE 4: Schematic diagram of blast furnace system.

(Kg/t), blast humidity  $x_5$  ( $g/m^3$ ), and gas volume of bosh  $x_6$  ( $m^3/min$ ).

Figure 6 displays the modeling data sets collected from the 9 to 21 October 2013. To better exhibit the performance of ELM based prediction model, optimal learning parameters

need to be made. For the sake of simplicity, we mainly discuss the reason of selecting the optimal number of hidden nodes for the ELM algorithm. The optimal number of hidden units is selected as the one which results in the lowest validation error. Through experiments analysis, the optimal number of

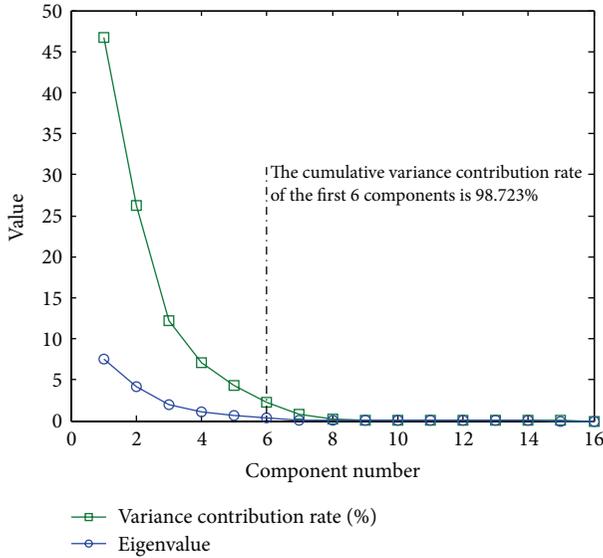


FIGURE 5: Eigenvalue and variance contribution rate of each component.

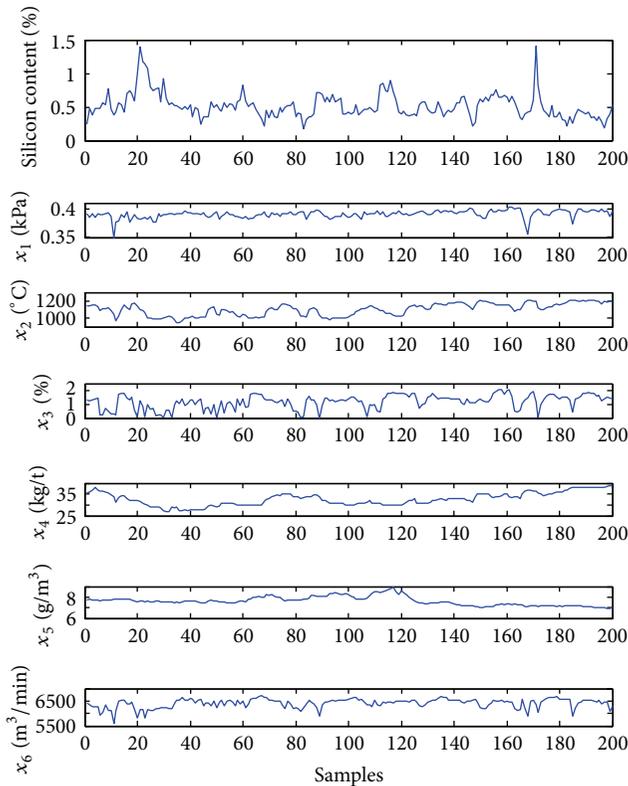


FIGURE 6: Data sets for prediction modeling.

hidden nodes with sigmoidal function is set as  $\tilde{N} = 25$ . The corresponding modeling result of the developed ELM model with self-feedback structure is shown in Figure 7, where the good modeling accuracy with practical data has been demonstrated.

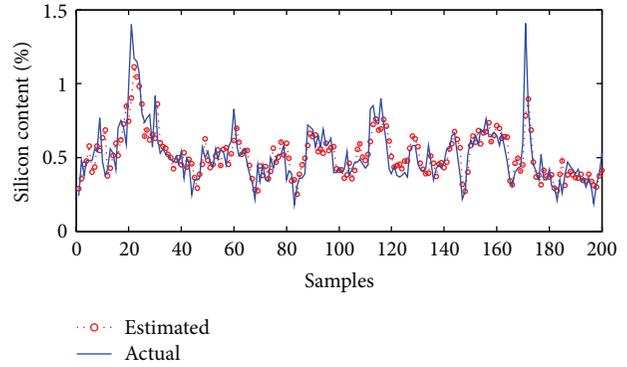


FIGURE 7: Modeling results with the proposed method.

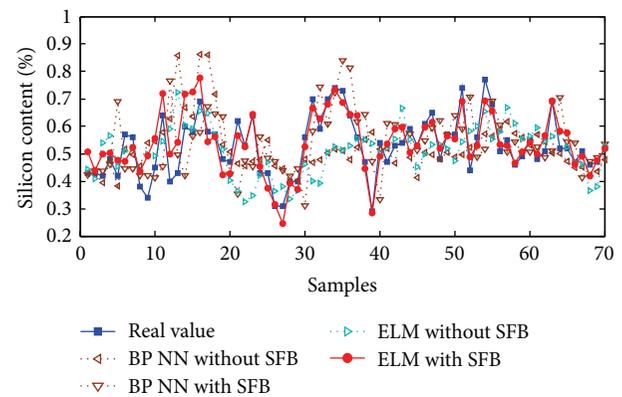


FIGURE 8: Estimation results of molten iron silicon content with different models.

**5.2. Experiments Results.** The developed ELM based prediction model has been tested on 2<sup>nd</sup> blast furnace in Liuzhou Steel of China for quite a long time. Figure 8 shows the estimated results using the proposed modeling method for predicting [Si], where the figure compares the predicted trend with the actual one. Moreover, in order to show the superiority of the proposed method more intuitively, comparisons with various popular prediction models have been made. Here, back propagation neural network without self-feedback (SFB) connection (BP NN for short), BP NN with SFB connection, and traditional ELM without SFB connection have been chosen to conduct the prediction comparison on the same observations. Moreover, the *Levenberg-Marquardt* algorithm is used in traditional BP learning algorithm, which appears to be the fastest method for training moderate-sized feedforward neural networks [16]. From Figure 8, it can be seen that the proposed model has the best estimation performance among all the developed prediction models. For example, it results in the best estimation trend and accuracy, and the shapes of the estimated curve values match the measured ones very well and better than that with other three methods.

It is well known that a good model should have its estimated error autocorrelation close to a white noise. So, in this text, we draw the autocorrelation function of estimating

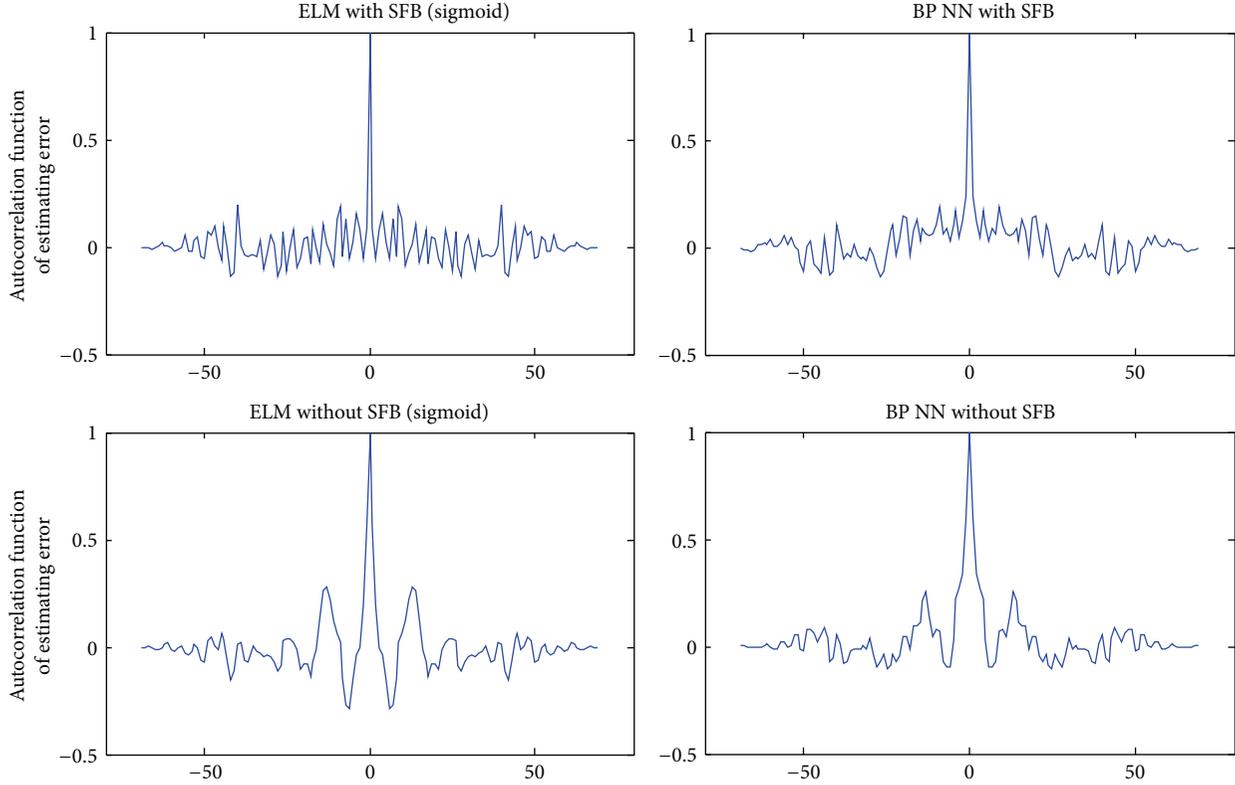


FIGURE 9: Autocorrelation function of estimating error of different models.

error of different models as shown in Figure 9. It can be seen from these figures that the autocorrelation results of algorithm like BP NN without SFB connection and ELM (sigmoid) without SFB connection are much worse than that with a SFB structure, respectively. Although one can obtain that the measuring error autocorrelations of the proposed ELM with SFB connection and BP NN with SFB connection are all satisfactory and close to the shape of the white noise here, the above estimation result (as shown in Figure 8) confirmed the effectiveness and superiority of the proposed method in predicting accuracy.

The estimation and generalization performance of the developed models can be further evaluated quantitatively by calculating the validation accuracy on the testing data set using the standard statistical measures, such as the *root mean squared error* (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{M} \sum_{i=1}^M (\tilde{y}_i - y_i)^2} \quad (13)$$

and the *standard deviation*  $\delta$  of estimation error  $|\tilde{y}_i - y_i|$

$$\delta = \sqrt{\frac{1}{M} \sum_{i=1}^M \left( |\tilde{y}_i - y_i| - \frac{1}{M} \sum_{j=1}^M |\tilde{y}_j - y_j| \right)^2}, \quad (14)$$

where  $M$  stands for the number of data points in the time series to be estimated,  $y_i$  is the actual value of time series, and  $\tilde{y}_i$  is the estimated value at time  $i$  by the prediction model.

TABLE 3: Some data statistics of each algorithm.

| Algorithm           | $\delta$<br>(testing) | RMSE<br>(testing) | Time (seconds)  |                 |
|---------------------|-----------------------|-------------------|-----------------|-----------------|
|                     |                       |                   | Training        | Testing         |
| BP NN without SFB   | 0.0827                | 0.1293            | 0.849955        | 0.013285        |
| BP NN with SFB      | 0.0804                | 0.1272            | 0.926292        | 0.013828        |
| ELM without SFB     | 0.0770                | 0.1187            | 0.000678        | 0.000072        |
| <b>ELM with SFB</b> | <b>0.0355</b>         | <b>0.1106</b>     | <b>0.000546</b> | <b>0.000089</b> |

Table 3 shows the calculated RMSE,  $\delta$ , and consuming time in training or testing procedure using different methods, respectively. It can be seen from this table that the ELM with feedback connection presented in this paper obtains a much less RMSE and  $\delta$  than other contrastive model algorithms. Moreover, the proposed model (sigmoid) spent the smallest 0.000546 s CPU time for training and an even faster time 0.000089 s for testing and obtained a very reasonable result. Through the analysis, it can also be seen that, no matter what kinds of algorithm are used, both training and testing results of a model with dynamic feedback are much better than the model without feedback structure when other conditions are the same, which confirm the effectiveness of our developed dynamic self-feedback model structure.

Moreover, the results of practical application indicate that the performance of the developed model is superior to other models and can overcome the problem of “over fitting” excellently. And the gap between every training and

testing is small, which enhances the reliability of the proposed method. The method can also overcome blindness of predefined parameters selection of conventional algorithm; thus convenience is provided for the operators.

## 6. Conclusions

This paper proposed a data-driven modeling for prediction of molten iron silicon content using PCA and ELM with self-feedback structure. Unlike other methods used for silicon content prediction, the proposed method can predict silicon content more accurately with an extremely fast speed than conventional algorithm, which feed the need for real-time control. Apart from selecting the number of hidden nodes, no other control parameter has to be chosen; thus convenience is provided for the operators. Moreover, the modified ELM with self-feedback structure can overcome the static mapping limitation of traditional ELM and so can cope with dynamic time-series prediction problems very well. Performance of the proposed modified ELM based prediction model is compared with BP algorithm and different model structure on practical industrial data obtained from 2<sup>#</sup> BF in Liuzhou Steel Company of China. The accuracy can basically meet the requirements of actual operation.

## Conflict of Interests

The authors have declared that they have no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (61104084, 614730646, 61290323, and 61333007), the Fundamental Research Funds for the Central Universities (N130508002 and N130108001), the IAPI Fundamental Research Funds (2013ZCX02-09), and the 111 Project (B08015). The authors would like to thank the anonymous reviewers for their constructive comments and the editors for their efforts in editing and polishing the paper. The authors would also like to thank the Ironmaking Plant of Liuzhou Iron & Steel Group Co. in China for providing a lot of experimental support.

## References

- [1] L. Jian, C. H. Gao, and Z. H. Xia, "Constructing multiple kernel learning framework for blast furnace automation," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 763–777, 2012.
- [2] C. H. Gao, L. Jian, and S. H. Luo, "Modeling of the thermal state change of blast furnace hearth with support vector machines," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1134–1145, 2012.
- [3] W. Birk, O. Marklund, and A. Medvedev, "Video monitoring of pulverized coal injection in the blast furnace," *IEEE Transactions on Industry Applications*, vol. 38, no. 2, pp. 571–576, 2002.
- [4] H. Saxen, C. H. Gao, and Z. W. Gao, "Data-driven time discrete models for dynamic prediction of the hot metal silicon content in the blast furnace—a review," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2213–2225, 2013.
- [5] S. Ueda, S. Natsui, H. Nogami, J.-I. Yagi, and T. Ariyama, "Recent progress and future perspective on mathematical modeling of blast furnace," *ISIJ International*, vol. 50, no. 7, pp. 914–923, 2010.
- [6] M. Phadke and S. M. Wu, "Identification of multiinput-multioutput transfer function and noise model of a blast furnace from closed-loop data," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 944–951, 1974.
- [7] M. Castore, G. Gandolfi, S. Palella, and G. Taspardini, "Dynamic model for hot-metal Si prediction in blast-furnace control," in *Proceedings of the Developments Ironmaking Practice*, pp. 152–159, Iron and Steel Institute, London, UK, 1972.
- [8] Y. C. Chao, C. W. Su, and H. P. Huang, "The adaptive autoregressive models for the system dynamics and prediction of blast-furnace," *Chemical Engineering Communications*, vol. 44, pp. 309–330, 1986.
- [9] T. Bhattacharya, "Prediction of silicon content in blast furnace hot metal using partial least squares (PLS)," *ISIJ International*, vol. 45, no. 12, pp. 1943–1945, 2005.
- [10] J. Jiménez, J. Mochón, J. S. de Ayala, and F. Obeso, "Blast furnace hot metal temperature prediction through neural networks-based models," *ISIJ International*, vol. 44, no. 3, pp. 573–580, 2004.
- [11] H. Saxén and F. Pettersson, "Nonlinear prediction of the hot metal silicon content in the blast furnace," *ISIJ International*, vol. 47, no. 12, pp. 1732–1737, 2007.
- [12] J. Chen, "A predictive system for blast furnaces by integrating a neural network with qualitative analysis," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 1, pp. 77–85, 2001.
- [13] X. Tang, L. Zhuang, and C. Jiang, "Prediction of silicon content in hot metal using support vector regression based on chaos particle swarm optimization," *Expert Systems with Applications*, vol. 36, no. 9, pp. 11853–11857, 2009.
- [14] C. Gao, Q. Ge, and L. Jian, "Rule extraction from fuzzy-based blast furnace SVM multiclassifier for decision-making," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 586–596, 2014.
- [15] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, IEEE, July 2004.
- [17] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [18] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.
- [19] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [20] Y. Lan, Y. C. Soh, and G.-B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13–15, pp. 3391–3395, 2009.

- [21] R. Moreno, F. Corona, A. Lendasse, M. Graña, and L. S. Galvão, "Extreme learning machines for soybean classification in remote sensing hyperspectral images," *Neurocomputing*, vol. 128, pp. 207–216, 2014.
- [22] Y. J. Sun, Y. Yuan, and G. R. Wang, "Extreme learning machine for classification over uncertain data," *Neurocomputing*, vol. 128, pp. 500–506, 2014.
- [23] R. Savitha, S. Suresh, and N. Sundararajan, "Fast learning circular complex-valued extreme learning machine (CC-ELM) for real-valued classification problems," *Information Sciences*, vol. 187, pp. 277–290, 2012.
- [24] Q. Yu, Y. Miche, E. Séverin, and A. Lendasse, "Bankruptcy prediction using extreme learning machine and financial expertise," *Neurocomputing*, vol. 128, pp. 296–302, 2014.
- [25] J. W. Cao, Z. P. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.
- [26] J. Cao and Z. Lin, "Bayesian signal detection with compressed measurements," *Information Sciences*, vol. 289, pp. 241–253, 2014.
- [27] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, June 2014.
- [28] J. Zhang, E. Martin, and A. J. Morris, "Fault detection and classification through multivariate statistical techniques," in *Proceedings of the American Control Conference (ACC '95)*, vol. 1, pp. 751–755, June 1995.
- [29] R. P. Good, D. Kost, and G. A. Cherry, "Introducing a unified PCA algorithm for model size reduction," *IEEE Transactions on Semiconductor Manufacturing*, vol. 23, no. 2, pp. 201–209, 2010.
- [30] J. Zhao, W. Wang, Y. Liu, and W. Pedrycz, "A two-stage online prediction method for a blast furnace gas system and its application," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 507–520, 2011.
- [31] L. Jian and C. Gao, "Binary coding SVMs for the multiclass problem of blast furnace system," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 9, pp. 3846–3856, 2013.
- [32] C. Gao, L. Jian, X. Liu, J. Chen, and Y. Sun, "Data-driven modeling based on volterra series for multidimensional blast furnace system," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2272–2283, 2011.

## Research Article

# An ELM Based Online Soft Sensing Approach for Alumina Concentration Detection

Sen Zhang, Xi Chen, and Yixin Yin

*School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China*

Correspondence should be addressed to Sen Zhang; zhangsen@ustb.edu.cn

Received 19 August 2014; Accepted 30 October 2014

Academic Editor: Yi Jin

Copyright © 2015 Sen Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The concentration of alumina in the electrolyte is of great significance during the production of aluminum; it may affect the stability of aluminum reduction cell and the current efficiency. However, the concentration of alumina is hard to be detected online because of the special circumstance in the aluminum reduction cell. At present, there is lack of fast and accurate soft sensing methods for alumina concentration and existing methods can not meet the needs for online measurement. In this paper, a novel soft sensing method based on a modified extreme learning machine (MELM) for online measurement of the alumina concentration is proposed. The modified ELM algorithm is based on the enhanced random search which is called incremental extreme learning machine in some references. It randomly chooses the input weights and analytically determines the output weights without manual intervention. The simulation results show that the approach can give more accurate estimations of alumina concentration with faster learning speed compared with other methods such as BP and SVM.

## 1. Introduction

In the industrial aluminum reduction cells, the stability of the alumina concentration is the key issue to maintain high efficiency during the production of aluminum. It is easy to lead to the occurrence of the so-called “anode effect” if the alumina content in the electrolyte becomes too low (e.g., 1%–1.5%). When it occurs, cell voltage rises abruptly to 30V–50V, which directly affects the energy balance of aluminum reduction cells. It is important to avoid the high alumina concentration because alumina-rich sludge will accumulate at the base of the cell and the cell operation can be seriously disrupted [1]. Therefore, how to detect the distribution of alumina concentration in the cell in real time is the key problem to control the production of aluminum. The site environment of aluminum reduction cell, which has the characteristics of large current, strong magnetic field, high temperature, high humidity, and a lot of dust, is very severe and complex. Aluminum reduction cell is a severe nonlinear, multi-input multioutput, slow time-variety, long time-delay, and coupled process. Therefore, it is a challenge to seek an online soft sensing method for alumina concentration. Numerous investigations have been carried out on the soft measurement method for alumina

concentration by the researchers. A prediction model based on wavelet neural network was proposed by Li et al. [2]. The prediction method based on linear regression and orthogonal transform is applied to improve the accuracy of the alumina concentration forecast by Lin et al. [3]. Yan and Liang proposed a predictive model of aluminum reduction cell based on LS-SVM [4]. Li et al. [5] proposed a new fuzzy expert control method based on smart identification, multicontrol mode, and decision-making mechanism to achieve the alumina concentration prediction and real time control. The GM(1, 1) model is introduced into the aluminum concentration estimate by Zhang et al. [6]. However, the computational burden of the above nonlinear predictive models is still large when the dimension of input variable increases; the learning speed and accuracy of networks are in general far slower and can not meet the requirement of real time detection.

Figure 1 shows the experimental environment in the ZunYi aluminium electrolysis factory in China. In industries, there often exist some crucial variables that can not be directly measured online due to the fact that no sensor is available in the certain complex environment or due to its high cost. Soft sensing technology is a way to solve the problems. The soft sensing technology is widely used and becomes



FIGURE 1: Experimental environment.

one of the important developing directions of surveying and processing control area. In practice, the application and research of this technology have gotten more extensions and many related technologies based on it have emerged [7–10].

The main thought of the soft sensing technology is to build one model which uses variables that could be directly online measured as input and thus the variables to be estimated as output, to use many kinds of complex calculating and evaluation and to get the values of detecting variables by computer software [11–14]. So the main problem of soft sensing technology is how to build the relation model between detecting variables and other easy getting variables. By now, there are many methods to build the models, and many methods are in the trend of intersecting and mixing together. In all the methods, the artificial intelligence method is used often, such as the method based on model identification, the method based on artificial neural network, or the method based on fuzzy set theory. Some professors and scholars have done some research about the technology, such as global asymptotic stability of neural networks with multiple time-varying delays and fuzzy model-based robust networked control for a class of nonlinear systems proposed by Zhang et al. [15, 16], weighted least squares support vector machines, robustness and sparse approximation, proposed by Suykens et al. [17], Artificial Neural Network in process engineering proposed by Willis et al. [18], and the modified extreme learning machine method by Cao et al. [19]. Soft sensing technology is applied to various fields; Wang and Ren proposed soft sensing method for wastewater treatment based on BP neural network [20]. Rolling bearing fault detection based on the Teager Energy Operator and Elman Neural Network was proposed by Liu et al. [21].

So far, the widely used method of determining alumina concentration in the industrial factory is to use the spectrometer to analyze the sampled electrolyte. It is an offline method. The improved ELM method is applied to build up the soft sensing online detection approach of alumina concentration in this paper. This ELM algorithm tends to provide the best generalization performance at extremely fast learning speed. The experimental result shows that the new method can produce the best generalization performance and learn much faster than the traditional prediction models.

The following sections are organized as follows. Section 2 shows the theory of the extreme learning machine. Section 3 proposes the modified ELM algorithm. Data selection and

data preprocessing are shown in Section 4. Section 5 gives the simulation results. Section 6 summarizes the conclusions.

## 2. The Theory of Extreme Learning Machine

Huang proposed extreme learning machine (ELM) algorithm; ELM was originally proposed for the single-hidden-layer feedforward neural networks (SLFNs) and then it extends to the generalized SLFNs. ELM can randomly generate the input weights and the bias of hidden nodes. It uses the theory of least squares to get the output weights. The learning speed of ELM can be thousands of times faster than traditional feedforward network learning algorithms like backpropagation (BP) algorithm while obtaining better generalization performance and smaller training error [22, 23]. Figure 2 shows the single-hidden-layer feedforward network (SLFN) architecture.

Considering there are  $N$  arbitrary distinct samples  $(X_i, T_i)$ , when  $X_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n$  and  $T_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^m$ , standard SLFNs with  $L$  hidden neurons and activation function  $h(x)$  are mathematically modeled as

$$\sum_{i=1}^L \beta_i h(W_i \cdot X_j + b_i) = O_j, \quad j = 1, 2, \dots, N, \quad (1)$$

where  $W_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$  is the weight vector connecting the  $i$ th hidden neuron and the input neurons,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the weight vector connecting the  $i$ th hidden neuron and the output neurons, and  $b_i$  is the threshold of the  $i$ th hidden neuron.

The fact that standard SLFNs with  $L$  hidden neurons with activation function  $h(x)$  can approximate these  $N$  samples with zero error means that  $\sum_{j=1}^L \|O_j - T_j\| = 0$ ; there exist  $\beta_i$ ,  $W_i$ , and  $b_i$  such that

$$\sum_{i=1}^L \beta_i h(W_i \cdot X_j + b_i) = T_j, \quad j = 1, 2, \dots, N. \quad (2)$$

The above  $N$  equations can be written compactly as

$$H\beta = T, \quad (3)$$

where

$$H(W_1, \dots, W_L, b_1, \dots, b_L, X_1, \dots, X_N) = \begin{pmatrix} h(W_1 \cdot X_1 + b_1) & \cdots & h(W_L \cdot X_1 + b_L) \\ \vdots & \ddots & \vdots \\ h(W_1 \cdot X_N + b_1) & \cdots & h(W_L \cdot X_N + b_L) \end{pmatrix}_{N \times L}, \quad (4)$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{pmatrix}_{L \times m}, \quad T = \begin{pmatrix} t_1^T \\ \vdots \\ t_L^T \end{pmatrix}_{N \times m}.$$

$H$  is called the hidden layer output matrix of the neural network. ELM is to minimize the training error as well as the norm of the output weight. Minimize  $\|H\beta - T\|^2$  and  $\|\beta\|$ .

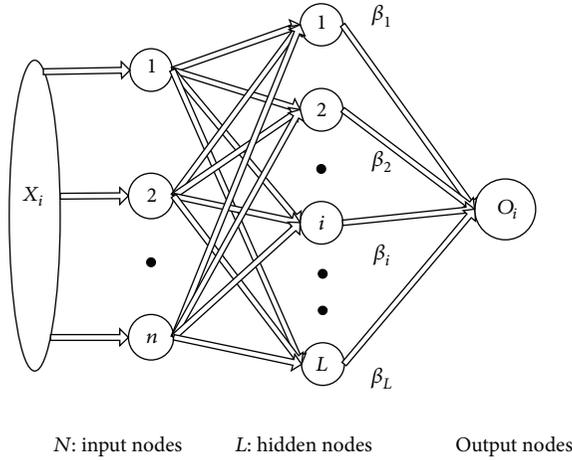


FIGURE 2: SLFN network architecture.

The minimal norm least squares method instead of the standard optimization method was used in the original implementation of ELM:

$$\hat{\beta} = H^T T, \quad (5)$$

where  $H^T$  is the Moore-Penrose generalized inverse of  $H$ . Several methods can be used to calculate the  $H^T$ ; these methods may include orthogonal projection, orthogonalization method, iterative method, and singular value decomposition (SVD), and ELM algorithm makes use of SVD, where  $H^T = (H^T H)^{-1} H^T$ , so

$$\hat{\beta} = (H^T H)^{-1} H^T T. \quad (6)$$

The algorithm ELM can be summarized as Three-Step Learning Model. Given a training set  $\aleph = \{(X_i, T_i) \mid X_i \in R^n, T_i \in R^m, i = 1, 2, \dots, N\}$ , an activation function  $h(x)$ , and the hidden neuron number  $L$ , we have the following steps.

*Step 1.* Assign arbitrary input weights  $W_i$  and bias of hidden layer nodes  $b_i$ ,  $i = 1, 2, \dots, L$ .

*Step 2.* Calculate the hidden layer output matrix  $H$ .

*Step 3.* Calculate the output weights  $\hat{\beta}$ .

### 3. The Modified ELM Algorithm

*3.1. Ridge Regression Based ELM Algorithm.* Ridge Regression, which was proposed by Horel and Kennard in 1970, proposes the idea of adding a small positive number on the main diagonal of the design matrix. Considering ELM algorithm, the estimation of  $\hat{\beta}^*$  can be obtained by employing the following formula:

$$\hat{\beta}^* = (H^T H + kI)^{-1} H^T T, \quad (7)$$

where  $k$  is the ridge parameter. Superficially, it is possible to get the inversion of design matrix  $H^T H$ . The following is the new results of (8) and (9):

$$\begin{aligned} \hat{\beta}^* &= \frac{\sum_{i=1}^{\bar{N}} H_i^T T_i}{\sum_{i=1}^{\bar{N}} H_i^2 + k} = \frac{\sum_{i=1}^{\bar{N}} H_i^T (H_i \beta_i + \varepsilon_i)}{\sum_{i=1}^{\bar{N}} H_i^2 + k} \\ &= \frac{\sum_{i=1}^{\bar{N}} H_i^T \beta_i}{\sum_{i=1}^{\bar{N}} H_i^2 + k} + \frac{\sum_{i=1}^{\bar{N}} H_i^T \varepsilon_i}{\sum_{i=1}^{\bar{N}} H_i^2 + k}, \end{aligned} \quad (8)$$

$$(1) E(\hat{\beta}^*) = \frac{\sum_{i=1}^{\bar{N}} H_i^2}{\sum_{i=1}^{\bar{N}} H_i^2 + k} \beta \neq \beta,$$

which denotes that Ridge Regression is biased.

Consider

$$(2) V(\hat{\beta}^*) = \frac{\sigma^2 \sum_{i=1}^{\bar{N}} H_i^2}{\left(\sum_{i=1}^{\bar{N}} H_i^2 + k\right)^2} = \frac{\sigma^2 \sum_{i=1}^{\bar{N}} \lambda_i}{\left(\sum_{i=1}^{\bar{N}} \lambda_i + k\right)^2} < V(\hat{\beta}), \quad (9)$$

which denotes that Ridge Regression makes the estimation more stable.

Consider

$$\begin{aligned} (3) \text{MSE}(\hat{\beta}^*) &= \frac{1}{\bar{N}} \\ &\times \left( \frac{\sigma^2 \sum_{i=1}^{\bar{N}} H_i^2 + \beta^2 k^2}{\left(\sum_{i=1}^{\bar{N}} H_i^2 + k\right)^2} \right. \\ &= \frac{\sigma^2 \sum_{i=1}^{\bar{N}} \lambda_i}{\left(\sum_{i=1}^{\bar{N}} \lambda_i + k\right)^2} + \frac{\beta^2 k^2}{\left(\sum_{i=1}^{\bar{N}} \lambda_i + k\right)^2} \left. \right) \\ &= \frac{1}{\bar{N}} (\gamma_1(k) + \gamma_2(k)). \end{aligned} \quad (10)$$

Hoerl and Kennard had proved that Ridge Regression has less mean square error than the ordinary regression under the proper ridge parameter. It is as follows:

$$\begin{aligned} \frac{d\gamma_1(k)}{dk} &= -\frac{2\sigma^2 \sum_{i=1}^{\bar{N}} \lambda_i}{\left(\sum_{i=1}^{\bar{N}} \lambda_i + k\right)^3}, \\ \frac{d\gamma_2(k)}{dk} &= \frac{2\beta^2 k \sum_{i=1}^{\bar{N}} \lambda_i}{\left(\sum_{i=1}^{\bar{N}} \lambda_i + k\right)^3}. \end{aligned} \quad (11)$$

So, when  $k \rightarrow 0^+$ ,

$$\lim_{k \rightarrow 0^+} \frac{d\gamma_1(k)}{dk} = -\frac{2\sigma^2}{\left(\sum_{i=1}^{\bar{N}} \lambda_i\right)^2} < 0, \quad (12)$$

$$\lim_{k \rightarrow 0^+} \frac{d\gamma_2(k)}{dk} = 0.$$

From the above equations,  $\text{MSE}(\hat{\beta}^*)$  is an increasing function of  $k$  when  $k \in (0, \delta)$ , where  $(d\gamma_1(k)/dk)|_{k=\delta} = (d\gamma_2(k)/dk)|_{k=\delta}$ . Therefore, the selection of parameter  $k$  is essential to the performance of Ridge Regression. In our ER-ELM algorithm, a method to determine the ridge parameter proposed by Huang [24] is used:

$$k = \frac{\hat{\sigma}^2}{\hat{\beta}}, \quad (13)$$

where  $\hat{\beta}$  is the ordinary ELM algorithm estimation and  $\hat{\sigma}^2 = \sum_{i=1}^N (y_i - \hat{\beta}x_i)^2 / \nu$ ,  $\nu = n - 1$ .

**3.2. To Select the Number of Hidden Nodes Based on the Improved ELM.** The Error Minimized Extreme Learning Machine algorithm starts from a small size of ELM hidden layer and adds random hidden node (nodes) to the hidden layer, while the output weights are updated incrementally.

Suppose a SLFN,  $H_1 = H(w_1, \dots, w_{l_0}, b_1, \dots, b_{l_0}, k_1)$ , denotes the hidden layer output matrix with  $l_0$  hidden nodes and ridge parameter  $k_1$  calculated by (19). Considering the poor performance due to lower number of hidden nodes, additional  $l$  hidden nodes are added to the SLFN. A new hidden layer output matrix  $H_2$  is composed of  $H_1$  and other  $l$  extra hidden nodes as

$$H_2 = [H_1, H], \quad (14)$$

where  $H = \begin{pmatrix} G(w_{l_0+1}, b_{l_0+1}, x_1, k_2) & \dots & G(w_{l_1}, b_{l_1}, x_1, k_2) \\ \vdots & \ddots & \vdots \\ G(w_{l_0+1}, b_{l_0+1}, x_N, k_2) & \dots & G(w_{l_1}, b_{l_1}, x_N, k_2) \end{pmatrix}_{N \times (l_1 - l_0)}$ ,  $l_1 -$

$l_0 = l$ , and  $k_2$  is ridge parameter of  $H^T H$ .

Huang had proved  $E(H_2) = \min \|H_2 \beta_2 - T\| \leq E(H_1) = \min \|H_1 \beta_1 - T\|$ , where  $E(H)$  denotes the output error function of SLFNs. We set a stopping criterion for the iterative algorithm as follows:

$$|E(H_{n+1}) - E(H_n)| < \varepsilon, \quad (15)$$

where  $\varepsilon > 0$  is called the target error.

Consider

$$\begin{aligned} H_2^+ &= (H_2^T H_2 + K)^{-1} H_2^T \\ &= \left( \begin{bmatrix} H_1^T \\ H^T \end{bmatrix} [H_1 \ H] + \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} \right)^{-1} \begin{bmatrix} H_1^T \\ H^T \end{bmatrix}, \end{aligned} \quad (16)$$

where  $K = kI$ .

In order to facilitate the calculation, the inversion is substituted as follows:

$$\begin{aligned} \Phi &= \begin{bmatrix} \varphi_{11} & \varphi_{12} \\ \varphi_{21} & \varphi_{22} \end{bmatrix} \\ &= \left( \begin{bmatrix} H_1^T \\ H^T \end{bmatrix} [H_1 \ H] + \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} \right)^{-1}, \end{aligned} \quad (17)$$

where

$$\begin{aligned} \varphi_{11} &= (H_1^T H_1 + K_1)^{-1} \\ &\quad + (H_1^T H_1 + K_1)^{-1} H_1^T H R^{-1} H^T H_1 (H_1^T H_1 + K_1)^{-1}, \\ \varphi_{12} &= -(H_1^T H_1 + K_1)^{-1} H_1^T H R^{-1}, \\ \varphi_{21} &= -R^{-1} H^T H_1 (H_1^T H_1 + K_1)^{-1}, \\ \varphi_{22} &= R^{-1}, \\ R &= H^T H + K_2 - H^T H_1 (H_1^T H_1 + K_1)^{-1} H_1^T H \\ &= H^T H + K_2 - H^T H_1 H_1^+ H. \end{aligned} \quad (18)$$

So

$$H_2^+ = \begin{bmatrix} U \\ D \end{bmatrix} = \begin{bmatrix} \varphi_{11} H_1^T + \varphi_{12} H^T \\ \varphi_{21} H_1^T + \varphi_{22} H^T \end{bmatrix}, \quad (19)$$

$$\begin{aligned} D &= R^{-1} H^T - R^{-1} H^T H_1 (H_1^T H_1 + K_1)^{-1} H_1^T \\ &= R^{-1} H^T - R^{-1} H^T H_1 H_1^+ \\ &= (H^T H + K_2 - H^T H_1 H_1^+ H)^{-1} H^T (I - H_1 H_1^+) \\ &= [H^T (I - H_1 H_1^+) H + K_2]^{-1} H^T (I - H_1 H_1^+) \\ &= (H^T M H + K_2)^{-1} H^T M, \end{aligned} \quad (20)$$

where  $M = I - H_1 H_1^+$ .

Similarly, we get  $U = H_1^+ - H_1^+ H D$ .

Now, a new hidden layer output matrix is obtained which has less output error. Then, we can update the output weight matrix based on the new hidden layer output matrix.

## 4. Data Selection and Preprocessing

In the aluminum production, through referring to relative documents and soliciting experts opinion, there are many factors that affect the alumina concentration, such as alumina feeding speed, cell voltage, series current, current of anode rod, voltage between anode rod and cathode bar, and bath temperature [25]. Existing predictive method of alumina concentration is to get an average alumina content in the aluminum reduction cell; the distribution of alumina concentration in the cell is not known yet. At present, people want

to know the distribution situation of alumina content in the cell in order to know the process of production better and control the production of aluminum more accurately. Our experimental aluminum reduction cell has 24 anode rods; we can get the current signal of 24 anode rods, voltage signal between anode rods and cathode bars, and alumina concentration signal below the 24 anode rods. Through experiment and analysis, we find that the voltage signal between anode rod and cathode bar can reflect the alumina concentration below the anode rod more accurately. So we can use the voltage signal between anode rod and cathode bar in the aluminum reduction cell to reflect the distribution of alumina concentration. To use the ELM algorithm, we decide to select voltage between anode rod and cathode bar variable as model input according to experiments, and output variable is alumina concentration in the electrolyte below the anode rod.

It is necessary to do the data denoising preprocessing before using the algorithm, because the process of data collection may introduce noise, where data collection is affected by interference and influence of all kinds of noise signal. At present, there are two denoising methods, the traditional filtering method and the wavelet denoising method. Traditional denoising method is based on Fourier analysis and can always be used in the environment where signal and noise are very small. While wavelet analysis is known as the “microscope” of mathematical analysis, it is a time-frequency analysis method of signal, with the characteristics of multiresolution analysis [26]. Different signals may choose the different denoising methods. Through experiment and comparison, we decide to use the wavelet denoising method in view of the voltage signal between anode rod and cathode bar.

## 5. Simulative Results of Alumina Concentration

In this paper, we select voltage between anode rod and cathode bar as model input and alumina concentration in the electrolyte below the anode rod as model output.

First, ELM algorithm is applied to build soft sensing method model of alumina concentration. In general, active functions play an important role in computing of neural networks. Widely used active functions in the ELM algorithm are *sig*, *sin*, *hardlim*, and *radbas*. Through comparison, we find that the *sig* function has more outstanding performance than *sin*, *hardlim*, and *radbas*. So we apply *sig* function as active function in the ELM algorithm. In order to make comparison, we also use the BP and SVM algorithm to build soft sensing models of alumina concentration [27, 28]. The BP parameters are chosen as 17 hidden layer neurons and 1 output layer neuron and transfer function of hidden layer is tan-sigmoid and transfer function of output layer is *linear*.

Data in this paper came from a 350 kA prebaked aluminum reduction cell in the ZunYi aluminium electrolysis factory; Figure 3 shows experimental aluminum reduction cell and anode rod. We got the data of voltage between anode rod and cathode bar through the voltage measurement instruments. At the same time, we obtained the electrolyte below the experimental anode rod in the experimental

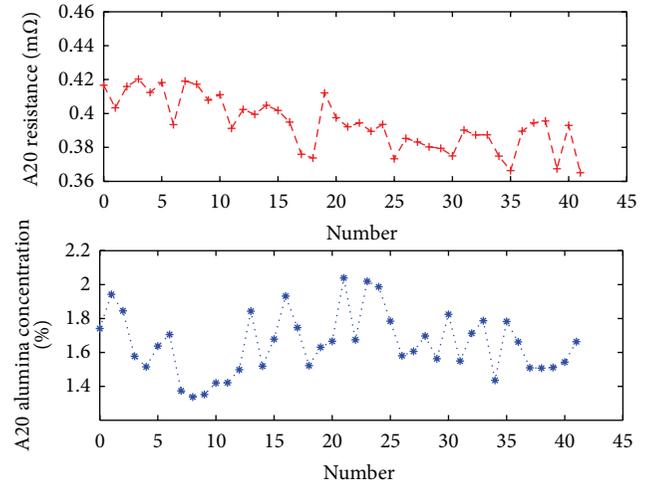


FIGURE 3: The voltage between number 20 anode rod and cathode bar and the alumina concentration in the electrolyte below the anode rod.

aluminum reduction cell, and then we used the X fluorescence spectrometer in the laboratory to analyze the alumina concentration. Figure 4 shows the sampled electrolyte. We got the experimental data through this method. We select 100 pairs as samples to construct sample set  $(X_i, T_i)$ , where  $X_i \in R$  denotes voltage between anode rod and cathode bar and  $T_i \in R$  denotes alumina concentration obtained at the same time with the voltage signal.

The ELM, BP, and SVM soft sensing models are trained by the same sample set. All the data are preprocessed before training and simulating in the algorithm. For the purpose of comparing the three models quantitatively, we substitute ten  $X_i$  of new sample (out of the sample set) into ELM, BP, and SVM models, respectively, to calculate corresponding  $\hat{T}_i$ ; simulation results of ELM model are shown in Figure 5; the alumina concentration results of three models are shown in Table 1. To see the results more clearly, the actual alumina concentration values and simulation results are displayed under the same axis which is shown in Figure 6. There are five performance indicators to measure the quality of algorithm: Training Time, Testing Time, Training RMSE (root mean square error), Testing RMSE, and Average Relative Error (ARE), where

$$\text{ARE} = \frac{1}{10} \sum_{i=1}^{10} \frac{|T_i - \hat{T}_i|}{T_i} \times 100\%. \quad (21)$$

The simulation results of five performance indicators are shown in Table 2.

From Figure 7, the horizontal axis denotes the values of voltage between anode rod and cathode bar and the vertical axis denotes the alumina concentration below the experimental anode rod. The simulation results of ELM approach the sample data in a certain range. As is shown in Table 1 and Figure 8, the values of output of ELM model are closer to the actual values of alumina concentration. From Table 2, Training Time of ELM model is 0 s whose

TABLE I: Simulation alumina concentration results of three models.

|                   | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     |
|-------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Actual value (%)  | 1.94   | 2.37   | 2.03   | 1.95   | 2.16   | 2.02   | 1.99   | 1.94   | 2.64   | 2.94   |
| Output of ELM (%) | 1.8164 | 2.0641 | 2.0739 | 2.1544 | 1.9423 | 1.9065 | 1.9166 | 2.047  | 2.6761 | 2.6443 |
| Output of BP (%)  | 1.7551 | 2.015  | 1.9953 | 2.4558 | 1.9594 | 1.9047 | 1.9510 | 2.0183 | 3.0956 | 2.9993 |
| Output of SVM (%) | 1.9176 | 2.003  | 2.006  | 2.0917 | 1.8828 | 1.8834 | 1.8979 | 1.8833 | 2.301  | 2.3042 |

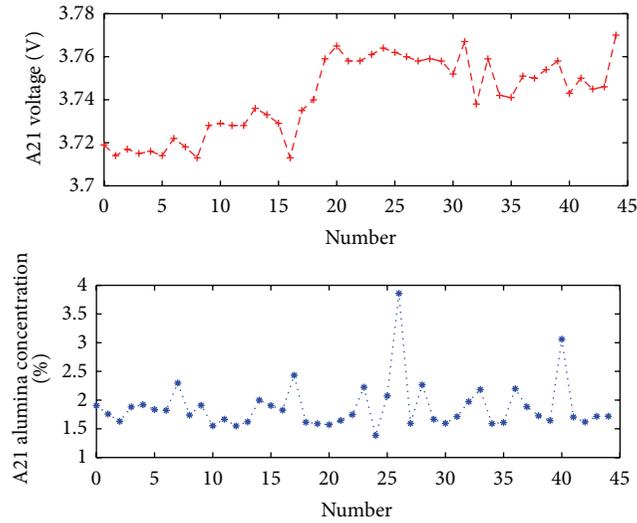


FIGURE 4: The voltage between number 21 anode rod and cathode bar and the alumina concentration in the electrolyte below the anode rod.



FIGURE 5: Experimental aluminum reduction cell and anode rod.

Training RMSE is 0.0926% and Testing RMSE is 0.1784%, while Training Time of BP model is 28.3281 s whose Training RMSE is 0.1698% and Testing RMSE is 0.2626%. In the SVM model, Training Time is 0.0156 s and Training RMSE and Testing RMSE are 0.2635% and 0.2797%. In terms of Average Relative Error (ARE) performance indicator, ELM has the smallest ARE (6.78%) in all of the three algorithms. It is clear that SVM model has faster learning speed and less Average Relative Error than BP model, but SVM model is slower and less accurate than ELM model. So the soft sensing measurement model based on ELM algorithm has better performance than BP model and SVM model.



FIGURE 6: Sampled electrolyte.

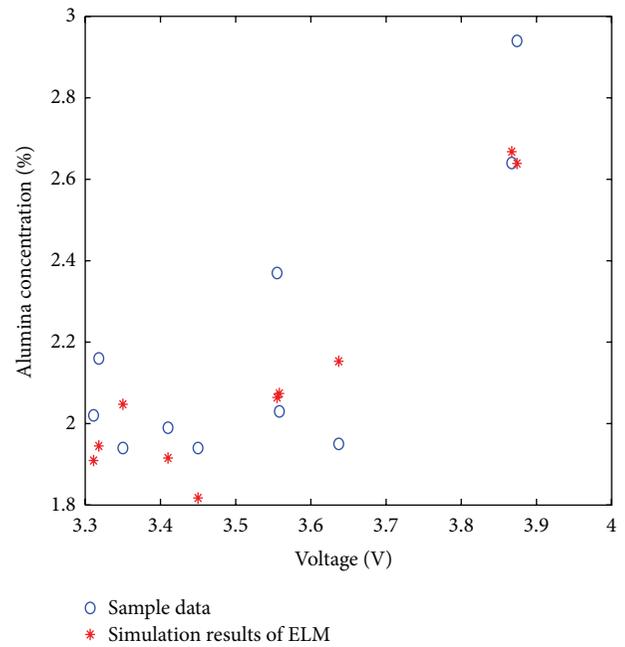


FIGURE 7: Simulation result of ELM model.

## 6. Conclusions

Alumina concentration is very important in the aluminum electrolysis, which may affect the performance of aluminum reduction cell. It is difficult to measure the alumina content due to the complicated environment of aluminum reduction cell. This paper proposes a novel soft sensing method of alumina concentration in the electrolyte based on extreme learning machine (ELM) and builds the relationship between

TABLE 2: The comparison of performance indicators.

|     | Training Time (s) | Testing Time (s) | Training RMSE | Testing RMSE | ARE (%) |
|-----|-------------------|------------------|---------------|--------------|---------|
| ELM | 0                 | 0                | 0.0926        | 0.1784       | 6.78    |
| BP  | 28.3281           | 0.0156           | 0.1698        | 0.2626       | 9.17    |
| SVM | 0.0156            | 0                | 0.2635        | 0.2797       | 8.83    |

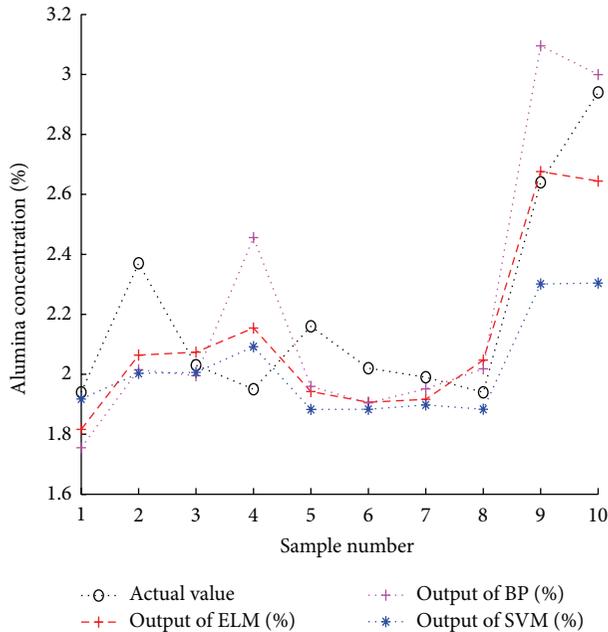


FIGURE 8: Actual alumina concentration values and simulation results.

alumina concentration and voltage between anode rod and cathode bar. Through the simulation results and comparison of BP model and SVM model, we can see the validity and advantage of the proposed method. This method is able to effectively achieve rapid and reliable estimation of alumina concentration in a relatively short time.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work has been supported by the National High Technology Research and Development Program (“863” Program) of China (Grant no. 2013AA040705).

## References

- [1] H. Vogt, “Effect of alumina concentration on the incipience of the anode effect in aluminum electrolysis,” *Journal of Applied Electrochemistry*, vol. 29, no. 7, pp. 779–788, 1999.
- [2] J. J. Li, C. D. Wang, and L. Ying, “Application research on neural network predictive control technology in Aluminum electrolysis process,” *Instrument Technique and Sensor*, vol. 8, pp. 91–93, 2011.
- [3] J. D. Lin, L. Li, and P. Zhang, “Research of predicting alumina concentration based on orthogonal transformation,” *Journal of Wuhan Institute of Technology*, vol. 32, pp. 9–13, 2010.
- [4] G. Yan and X. Liang, “Predictive models of aluminum reduction cell based on LS-SVM,” in *Proceedings of the International Conference on Digital Manufacturing and Automation (ICDMA '10)*, pp. 99–102, Changsha, China, December 2010.
- [5] J. Li, W.-G. Zhang, F.-Q. Ding, and Y.-X. Liu, “Fuzzy expert control method based on on-line intelligent identification and its application,” *Journal of Central South University of Technology*, vol. 35, no. 6, pp. 911–914, 2004.
- [6] H. Zhang, J. Li, W. Zhang, X. Chen, and Z. Zou, “Application of gray GM (1, 1) model to alumina concentration estimation in aluminum electrolysis,” *Chinese Journal of Scientific Instrument*, vol. 29, no. 4, pp. 883–887, 2008.
- [7] J. W. Cao, T. Chen, and J. Fan, “Fast online learning algorithm for landmark recognition based on BoW framework,” in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, Hangzhou, China, June 2014.
- [8] W. Xiao, P. Liu, W.-S. Soh, and G.-B. Huang, “Large scale wireless indoor localization by clustering and extreme learning machine,” in *Proceedings of the 15th International Conference on Information Fusion (FUSION '12)*, pp. 1609–1614, Singapore, September 2012.
- [9] W. Xiao, P. Liu, W. S. Soh, and Y. Jin, “Extreme learning machine for wireless indoor localization (poster),” in *Proceedings of the 11th International Conference on Information Processing in Sensor Networks (IPSN '12)*, pp. 101–102, Beijing, China, April 2012, (ISTP, EI).
- [10] W. Xiao, Y. Lu, J. Cui, and L. Ji, “Recognition of human stair ascent and descent activities based on extreme learning machine,” in *Proceedings of the 5th International Conference on Extreme Learning Machines (ELM '14)*, Marina Bay Sands, Singapore, 2014.
- [11] M. J. Arauzo-Bravo, J. M. Cano-Izquierdo, E. Gomez-Sanchez et al., “Automatization of a penicillin production process with soft sensors and an adaptive controller based on neuro fuzzy systems,” *Control Engineering Practice*, vol. 12, pp. 1073–1090, 2004.
- [12] A. J. de Assis and R. M. Filho, “Soft sensors development for on-line bioreactor state estimation,” *Computers and Chemical Engineering*, vol. 24, pp. 1099–1103, 2000.
- [13] F. L. Huang, “Thought of soft sensing and technology of soft Sensing,” *Journal of Metrology*, vol. 7, 2004.
- [14] J. Yu, “Soft sensing technology and its application,” *Journal of Automatic Instrument*, vol. 1, pp. 71–78, 2008.
- [15] H. Zhang, Z. Wang, and D. Liu, “Global asymptotic stability of recurrent neural networks with multiple time-varying delays,” *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 855–873, 2008.

- [16] H. G. Zhang, M. Li, J. Yang, and D. D. Yang, "Fuzzy model-based robust networked control for a class of nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 39, no. 2, pp. 437–447, 2009.
- [17] J. A. K. Suykens, J. de Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing*, vol. 48, pp. 85–105, 2002.
- [18] M. J. Willis, C. Di Massimo, G. A. Montague, M. T. Tham, and A. J. Morris, "Artificial neural networks in process engineering," *IEE Proceedings D*, vol. 138, no. 3, pp. 256–266, 1991.
- [19] J. W. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.
- [20] W.-L. Wang and M. Ren, "Soft-sensing method for wastewater treatment based on BP neural network," in *Proceedings of the 4th World Congress on Intelligent Control and Automation*, pp. 2330–2332, Shanghai, China, June 2002.
- [21] H. Liu, J. Wang, and C. Lu, "Rolling bearing fault detection based on the teager energy operator and elman neural network," *Mathematical Problems in Engineering*, vol. 2013, Article ID 498385, 10 pages, 2013.
- [22] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.
- [23] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [24] J. C. Huang, "Improving the estimation precision for a selected parameter in multiple regression analysis: an algebraic approach," *Economics Letters*, vol. 62, no. 3, pp. 261–264, 1999.
- [25] J. Li, Y. Huang, H. Wang, and Y. Liu, "Estimation model of alumina concentration for point-feeding aluminum reduction cells," in *Proceedings of the 123rd TMS Annual Meeting on Light Metals*, pp. 441–447, March 1994.
- [26] X. Zhang, D. Xu, and Z. Qi, "Study of wavelet denoising method Based on the modulus maxima domain," *Data Acquisition and Processing*, no. 9, pp. 315–318, 2003.
- [27] A. Meghlaoui, J. Thibault, R. T. Bui, L. Tikasz, and R. Santerre, "Neural networks for the identification of the aluminium electrolysis process," *Computers & Chemical Engineering*, vol. 22, no. 10, pp. 1419–1428, 1998.
- [28] J. A. K. Suykens, J. Vandewalle, and B. de Moor, "Optimal control by least squares support vector machines," *Neural Networks*, vol. 14, no. 1, pp. 23–35, 2001.

## Research Article

# Two-Dimensional Extreme Learning Machine

Bo Jia, Dong Li, Zhisong Pan, and Guyu Hu

College of Command Information System, PLA University of Science and Technology, Nanjing 210007, China

Correspondence should be addressed to Guyu Hu; huguyu@189.cn

Received 17 August 2014; Revised 5 November 2014; Accepted 6 November 2014

Academic Editor: Amaury Lendasse

Copyright © 2015 Bo Jia et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Extreme learning machine (ELM) has achieved wide attention due to faster learning speed compared with conventional neural network models like support vector machine (SVM) and back-propagation (BP) networks. However, like many other methods, ELM is originally proposed to handle vector pattern while nonvector patterns in real applications need to be explored, such as image data. We propose the two-dimensional extreme learning machine (2DELML) based on the very natural idea to deal with matrix data directly. Unlike original ELM which handles vectors, 2DELML take the matrices as input features without vectorization. Empirical studies on several real image datasets show the efficiency and effectiveness of the algorithm.

## 1. Introduction

Pattern representation is probably one of the basic problems in machine learning; almost all learning algorithms aim to build the mapping functions from the input to output. The output value of a learning model is always straightforward while different input representations could influence the results much. For statistical learning, the input pattern is commonly represented by a vector which contains the values belongs to corresponding features. Even though the original data is not sampled as vectors, there exists a standard preprocessing method named vectorization, which aims to transform the original data into vectors for the convenience of computation. Taking the face image, for example, each sample of a  $d_1$ -by- $d_2$  face image is always transformed into a  $d_1 \times d_2$ -length vector by concatenating all columns or rows, so that the sample can be processed by popular learning algorithms such as support vector machine (SVM) or artificial neural networks. *Input vectors* almost become another name for input samples, and some of them have discriminative ability which define the margin of largest separation are called support vectors in SVM [1].

On the one hand, vectorization helps the input data to fit in mature models as well as to accelerate computation procedure using popular linear algebra libraries. On the other hand, the drawbacks of vectorizing image data are obvious

from at least two aspects [2, 3]. (1) Structural or contextual information may be lost during the transformation due to the changes of relative position of the pixels, and the reason is quite intuitive. (2) Vectorization needs more parameters and thus leads to the curse of dimensionality. For example, in order to classify  $1024 \times 1024$  images by neural networks with 1000 hidden nodes, one need  $10^9$  parameters in the first layer. The feedforward computation can be slow.

Now look at the general class of mapping function adopted by many discriminative models, which take the sample vector as input and classification label or regression value as the output:

$$y = f \left( \sum_{i=1}^m \beta_i h_i(\mathbf{x}) \right), \quad (1)$$

where  $\mathbf{x} \in \mathbb{R}^d$  is the input vector and  $h_i(\mathbf{x})$  is the  $i$ th output value of the hidden layer in three-layer neural network, or the  $i$ th output value of other two-layer model such as least square regression and logistic regression.  $\beta_i$  is the parameter vector which connects  $h_i(\mathbf{x})$  and the final output value. In order to have a scalar output  $y$  easily, a linear or nonlinear transformation needs to be conducted on the input space; thus  $h_i(\mathbf{x})$  is sometimes regarded as point in the feature space. Function  $f(\cdot)$  controls the final output value according to

specific learning tasks. The definition of feature mapping function is

$$h_i(\mathbf{x}) = g(\mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i), \quad (2)$$

where  $\mathbf{w}_i \in \mathbb{R}^d$  is the weight vector that connects the input nodes and the  $i$ th hidden node in neural network models and  $\mathbf{b}_i$  is the bias of the  $i$ th hidden node in this case.  $g(\cdot)$  is probably a nonlinear continuous function. For linear regression models as well as back-propagation networks, the  $\mathbf{w}_i$  are the main parameters that need to be learned. The feature mapping stage here is a linear transformation, and the output of each hidden node is a linear combination of input units and corresponding weights.

Similar to vector case, the feature mapping function for the matrix pattern  $A \in \mathbb{R}^{d_1 \times d_2}$  looks differently as the following form [4]:

$$h_i(A) = g(\mathbf{u}_i^T A \mathbf{v}_i + \mathbf{b}_i), \quad (3)$$

where  $\mathbf{u}_i \in \mathbb{R}^{d_1}$  and  $\mathbf{v}_i \in \mathbb{R}^{d_2}$  are two weight vectors similar to  $\mathbf{w}_i$  in the vector pattern. This might be the simplest way to transform a matrix into a scalar using vector inner product similar to (2), since matrix-vector product is essentially sum of several vector inner products.

We can see that there are only  $d_1 + d_2$  parameters needed instead of  $d_1 \times d_2$  in (2) for each hidden node. From this point, using matrix pattern could reduce model complexity with fewer parameters, even if the original sample is not matrix as long as the vector can be recombination into matrix. Take the single layer feedforward neural network (SLFN), for example, here, as Figure 1 shows the differences between two input patterns: (a) needs  $d_1 + d_2$  nodes in the input layer while (b) just needs  $d_2$  for the same input sample.

As opposed to vector case learning methods, two-dimensional methods have been used on feature extraction as well as conventional learning models, in the last decade. Yang et al. [5] proposed two-dimensional principle component analysis (2DPCA) for image representation, which turned out to be advantageous over PCA in several aspects. Ye et al. [6] proposed two-dimensional linear discriminant analysis (2DLDA) which works with data in matrix representation and could overcome the singularity problem in conventional LDA. Wang et al. [3] provided a fully matrixed approach, applied in both feature extraction and classifier design, including his previous work [4] which proposed MatLSSVM, that is, least squares support vector machines (LS-SVM) based on matrix patterns and its fuzzy version. Empirical studies in these literatures showed that two-dimensional methods helped to improve classification performance and reduce the computational and space complexity, compared with the base models.

A more general representation pattern over matrix is tensor, which takes  $A \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_m}$  as the input. Tao et al. [7] described a supervised tensor learning framework and the alternating projection optimization to obtain the solution. Conventional models like SVM and Fisher discriminant analysis were contained in this framework. Possible solutions of tensor based ELM will be discussed later.

Inspired by the very natural idea to let ELM process matrices directly and matrix pattern related works [3], we propose the two-dimensional extreme learning machine (2DELm) in this paper, and our main contributions can be summarized as follows:

- (i) providing a simple method to process matrix pattern for SLFN;
- (ii) analyzing the random feature mapping from a probabilistic perspective for both ELM and 2DELm;
- (iii) comparing the proposed algorithm with original ELM on image datasets based on a statistical approach.

The remainder of this paper is organized as follows. Section 2 reviews the vector based ELM. Section 3 describes the 2DELm and related concepts, including a sparse version, kernels tricks, and tensor based ELM. We evaluate our methods on several image data in Section 4. Finally Section 5 concludes this paper.

## 2. Extreme Learning Machine: A Vector Case

Extreme learning machine [8] was proposed as an efficient learning algorithm for single hidden layer feedforward neural networks, which outperforms the gradient-based methods to learn the same architecture. The structure is also shown in Figure 1(a). According to a general principle for learning machine, that is, to minimize the empirical risk (ERM), ELM aims to reach the smallest training error by

$$\min_{\boldsymbol{\beta}} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\|_2^2, \quad (4)$$

where  $\mathbf{H}$  is defined as the hidden layer output matrix of  $N$  training samples,  $\boldsymbol{\beta}$  is output weights vector that connect hidden layer and output layer (with  $L$  hidden nodes), and  $\mathbf{y} \in \mathbb{R}^N$  is the target vector that contains real values for regression and class labels for classification:

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{pmatrix} = \begin{pmatrix} h_1(\mathbf{x}_1) & \dots & h_L(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \dots & h_L(\mathbf{x}_N) \end{pmatrix}, \quad (5)$$

where  $h_i(\mathbf{x}_j)$  is the output of the  $i$ th hidden node of the  $j$ th input vector and probably has the same form as (2).

The significant characteristic of ELM lies at the random choice of the weights  $\mathbf{w}$  that connect the input layer and hidden layer as well as the bias  $\mathbf{b}$  of hidden layer, which is different from traditional algorithms like back-propagation where all parameters need to be tuned. This makes the hidden layer output matrix by hand and only the output weights  $\boldsymbol{\beta}$  need to be learned. Under the ERM principle, the optimal solution  $\boldsymbol{\beta}^*$  to (3) can be analytically resolved as

$$\boldsymbol{\beta}^* = \mathbf{H}^\dagger \mathbf{y}, \quad (6)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ .

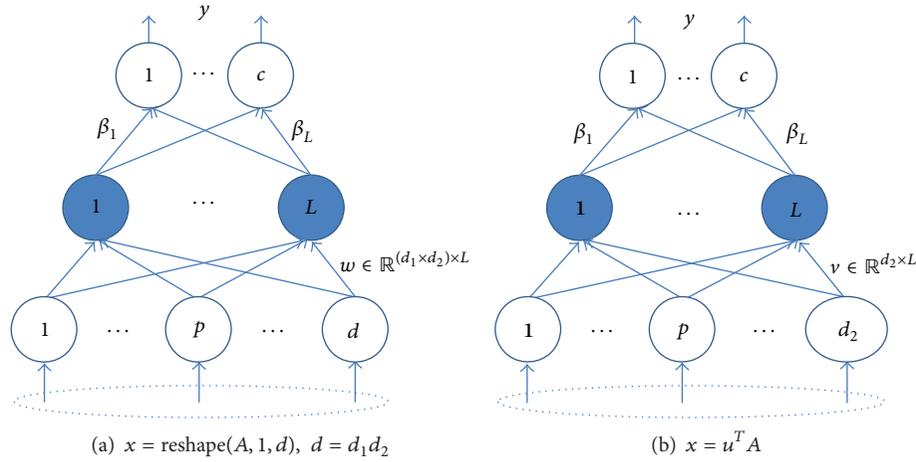


FIGURE 1: Two cases for single layer feedforward neural network on matrix pattern. (a) needs  $d_1 \times d_2$  nodes in the first layer while (b) just needs  $d_2$ .  $\text{reshape}(\cdot)$  here is the vectorization process.

The idea that hidden node parameters need not to be learned has been extended to many other models beyond neural network models like SVM, RBF networks, and so forth [9]. The simplicity of ELM has been also extended to form a unified framework, which mainly takes three steps as follows.

- (1) Randomly choose parameters at first layer of SLFN for feature mapping.
- (2) Use various activation function to generate new feature representations.
- (3) Fast solution of required parameters at last layer of SLFN.

Kernel tricks such as in SVM could be used in ELM to obtain more powerful classification ability [10, 11]. The fast solution in step (3) makes online learning and real time prediction possible. The whole procedure is also suitable for many other models in ensemble learning; the weights of multiple predictors can be determined by a similar way in (5). To be more specific, the last layer of SLFN can be viewed as a linear combination of multiple weak predictors to form a strong predictor, which is consistent with the design of ensemble learning.

Because of above properties, ELM and its variants have been widely used in many areas like face recognition [12], object recognition [13], large scale data analysis [14], network security [15], and so forth. Almost all these application examples deal with vector pattern, even if the objects are images. It is necessary to extend ELM to matrix pattern, so that we can use it in a more generalized form in practise.

### 3. Two-Dimensional ELM

**3.1. Basic Formulas.** The goals of 2DELm are to process matrix pattern directly, instead of vectorizing by concatenating all columns or rows at first. At the feature mapping stage, which is corresponding to the input layer and the hidden layer in SLFN architecture, each hidden node encode all original features of a sample somehow.

Assume activation function is sigmoid  $g(t) = 1/(1 + \exp(-t))$ , vector based ELM takes a linear combination of all features as input of activation function  $t$ , and the linear weights are randomly generated. Inspired by ELM, the first layer parameters which actually do feature mapping need not to be tuned in SLFN. In order to get random features in the hidden layer like ELM, we can randomly choose  $\mathbf{u}_i$ ,  $\mathbf{v}_i$ , and  $\mathbf{b}_i$  in (3), at the first layer in SLFN. As we mentioned,  $\mathbf{u}^T A \mathbf{v}$  might be the simplest way to transform a matrix  $A$  into a scalar using vector inner products. The entries in hidden layer output matrix  $(\mathbf{H})_{N \times L}$  of SLFN are formally defined as

$$\mathbf{H}_{ij} = g(\mathbf{u}_i^T A_j \mathbf{v}_i + \mathbf{b}_i), \quad (7)$$

where  $\mathbf{H}_{ij}$  is the output of the  $i$ th hidden node of the  $j$ th input matrix sample. Each hidden node thus gets all entries' information of matrix  $A_j$  while keeping various via different random weights  $\mathbf{u}_i$ ,  $\mathbf{v}_i$ , and  $\mathbf{b}_i$ .

For a complete learning model, we have random parameters  $\mathbf{U} \in \mathbb{R}^{d_1 \times L}$ ,  $\mathbf{V} \in \mathbb{R}^{d_2 \times L}$ , and bias  $\mathbf{b} \in \mathbb{R}^L$ .  $\mathbf{u}_i$  and  $\mathbf{v}_i$  in (7) are the  $i$ th row of  $\mathbf{U}$  and  $\mathbf{V}$ , respectively. Having the hidden layer output matrix by hand, the next thing would be the same as ELM: to solve the optimal weights by (6). With  $L$  hidden nodes, we can see that  $(d_1 + d_2) \times L$  input parameters are needed here while the number is  $(d_1 \times d_2) \times L$  after vectorization. Conversely, we could also conduct reformatting a vector into a matrix to reduce the parameters as long as the length of the vector is not a prime.

In order to get a stable solution, ridge regression [16] could be applied to solve  $\boldsymbol{\beta}$  for ELM [9, 17] as well as 2DELm. The corresponding objective function is defined as (8), and  $\lambda$  is the parameters to balance the loss and  $L_2$  regularizer. This problem can be analytically solved by  $\boldsymbol{\beta} = (\mathbf{H}^T \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H}^T \mathbf{y}$  where  $\mathbf{I}$  is the identity matrix

$$\min_{\boldsymbol{\beta}} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2. \quad (8)$$

The whole procedure of MatELM is illustrated in Algorithm 1. As we can see, using the same amount of hidden

**Input:** Training samples  $\{(A_i, y_i)\}, i = 1, \dots, N$   
 Samples  $A_i \in \mathbb{R}^{d_1 \times d_2}$ , labels  $y_i \in \{1, \dots, C\}$   
**Output:** Model parameters  $\beta$ .  
 (0) Determine the network architecture with  $d_2$  input nodes,  
 $L$  hidden nodes and  $C$  output nodes;  
 (1) Randomly choose input parameters  $\mathbf{U}, \mathbf{V}$  and bias  $\mathbf{b}$ ;  
 (2) Compute the hidden layer output matrix  $\mathbf{H}$  as in (7);  
 (3) Solve output parameters  $\beta$  by (6) or (8)

ALGORITHM 1: 2DELMM.

nodes, ELM and MatELM have the same training speed to compute  $\beta$  since they share the size of  $\mathbf{H}$  and  $\mathbf{y}$ . But in theory, the computational complexity to build  $\mathbf{H}$  is different:  $O(d_1 \times d_2)$  for ELM and  $O(d_1 + d_2)$  for 2DELMM. In practice the speed to compute  $\mathbf{H}$  also depends on how the original data are stored: 2DELMM tends to outperform ELM if samples are stored in matrices and vice versa.

*3.2. Further Discussion.* As mentioned in Section 1, tensor is a more general representation pattern over matrix pattern. The essence of learning model is to transform the input to considerable output, regarding tensor based ELM; we can extend it similar to 2DELMM. The most import step lies in step (2) of Algorithm 1, that is, to compute the hidden layer output matrix  $\mathbf{H}$ . For tensor pattern  $A \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_m}$ , we can define the entries in  $\mathbf{H}$  by

$$\mathbf{H}_{ij} = g\left(\left(\mathbf{u}_{(m-1)i}^T \left(\mathbf{u}_{(m-2)i}^T \cdots \left(\mathbf{u}_{1i}^T A_j\right)\right)\right)\right) \mathbf{u}_{mi} + \mathbf{b}_i, \quad (9)$$

where the weights  $\mathbf{u}_{ki} \in \mathbb{R}^{d_k}$  are randomly chosen. Once the hidden layer output matrix  $\mathbf{H}$  is ready, the rest of training is the same as ELM. The sparse weight vector  $\beta$  can be also obtained by  $L_1$  norm regularizer in tensor based ELM as well.

Castaño et al. [18] proposed PCA-ELM, a robust and pruned ELM based PCA, which aims to determine the hidden nodes in ELM with the information retrieved from principal components analysis of training data. PCA-ELM reduced the model parameters by taking low-dimension training data, which is different from our method. Explicit vectorization is needed in these methods; however the frameworks are not in contradiction with 2DELMM since the latter focuses on the pattern representation. In other words, PCA related techniques can be combined with the idea of 2DELMM in practice.

## 4. Experiments

In this section, we mainly compare 2DELMM and ELM on image datasets for multiclass classification. Assume the number of classes is  $C$ ; we transform the label vector  $y$  into ground truth matrix  $T_{N \times C}$  in both training and testing stage. The definition of entries of  $T$  is

$$T_{ij} = \begin{cases} 1 & \text{when } A_i \in j, \\ -1 & \text{otherwise.} \end{cases} \quad (10)$$

The primary solution of  $\beta_{L \times C}$  in all experiments is based on Moore-Penrose generalized inverse (we replace the  $y$  by  $T$  in (6)) since it needs only one user-defined parameter: the number of hidden nodes  $L$ . In practice, it is very time consuming to choose the parameter  $\lambda$  when using ridge regression in a wide range; moreover, in many cases, Moore-Penrose generalized inverse solution tends to be stable as well. At predicting stage, for each tested sample  $A$ , we use (11) to get the output vector  $\mathbf{t}$ , and then the index of largest entry of  $\mathbf{t}$  is taken as the label

$$\mathbf{t} = \beta \mathbf{h}(A), \quad (11)$$

where  $\mathbf{h}(A)$  is hidden layer output vector of sample  $A$ , sized by  $L$ -length, and each entry has the same form as (7).

*4.1. Data Description.* In order to show the effectiveness of 2DELMM, we get several popular image datasets, and the application background covers face recognition and other image classifications. In specific, there are five face databases and four OCR datasets in the following discussion, the data size and dimensions vary in wide range.

- (1) Yale database (<http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html>): the Yale database contains 165 images from 15 persons, and each person has 11 images. These images were various from facial expressions and lighting conditions: center-light, happy, left-light, sad, and so forth. Each image has size of  $32 \times 32$  or  $64 \times 64$ . Figure 2(a) shows five sample images from this database.
- (2) Face database (<http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html>): the ORL has 400 images belonging to 40 persons, and each person has 10 images. These images were taken from different times, and the lighting, facial expressions (open/closed eyes), and facial details (glasses or no glasses) are various. Moreover, all the images were taken in front of the same dark background with the individual in an upright position. Each image has size of  $32 \times 32$  or  $64 \times 64$ . Figure 2(b) shows five sample images from this database.
- (3) UMist faces (<http://www.cs.nyu.edu/~roweis/data.html>): there are 575 grayscale face images from 20 different people. These pictures were taken with the individuals having different side angle against the camera. Each image is  $112 \times 92$  size and is manually cropped by Graham and Allinson at UMist [19]. Figure 2(c) shows five sample images from this database.
- (4) Georgia tech face database ([http://www.anefian.com/research/face\\_reco.htm](http://www.anefian.com/research/face_reco.htm)): this database contains 750 images of 50 people, each person has 15. All people in the database are represented by 15 color JPEG images with cluttered background taken at resolution  $640 \times 480$  pixels. The database also provides coordinates of the face rectangle. Here we only use the grayscale images for classification. Figure 2(d) shows five sample images from this database.

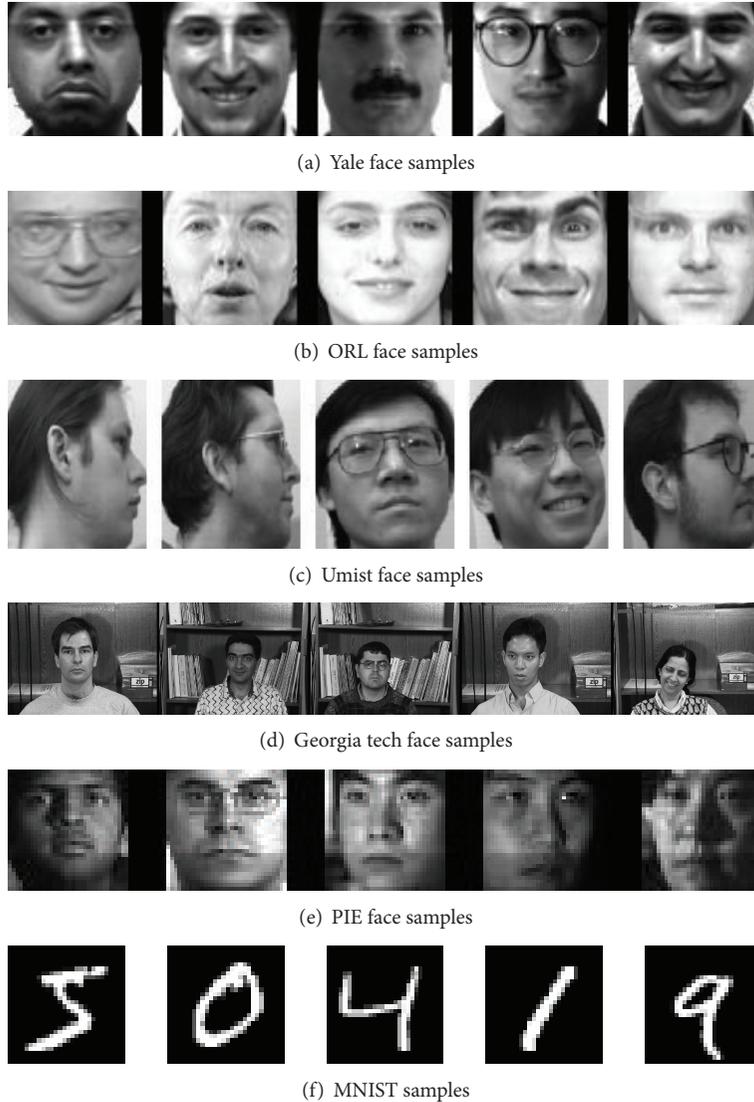


FIGURE 2: The samples from face database in the experiments.

- (5) PIE face: the CMU Pose, Illumination, and Expression (PIE) database, provided by [20]. There are 41,368 images of 68 people which were collected in 2000. These images were taken with each person under 13 different poses, 43 different illumination conditions, and with 4 different expressions. We get a subset containing 11,554 images in our experiment, and each has the size  $32 \times 32$ . Figure 2(e) shows five sample images from this database.
- (6) Letter, shuttle, and USPS datasets (<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html>) and MNIST Handwritten Digits (<http://www.cs.nyu.edu/~roweis/data.html>): these four OCR datasets were provided with separate training set and testing set. There are 7,291 training samples and 2,007 testing samples in USPS, and each has size  $16 \times 16$ . There are 15,000 training samples and 5,000 testing samples in letter, and each has size  $4 \times 4$ . There are

43,500 training samples and 14,500 testing samples in shuttle, and each has size  $3 \times 3$ . MNIST has 60,000 training samples and 10,000 testing samples, and each has size  $28 \times 28$ . Figure 2(f) shows five sample images from MNIST.

More details of these datasets are provided in Table 1. Similar to [4], we introduce the ratio  $R = (d_1 \times d_2) / (d_1 + d_2)$  to indicate the input parameters needed ratio for vector pattern versus matrix pattern. The last column of these tables indicates whether the training data and the testing data are provided separately.

4.2. *Experiment Settings.* The simulations of ELM ([http://www.ntu.edu.sg/home/egbhuang/elm\\_random\\_hidden\\_nodes.html](http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html)) and 2DELM (<https://github.com/fairmiracle/MatELM/>) on all datasets are carried out in Matlab 2013a environment running in Intel Core i5 CPU, with 16 GB memory.

TABLE 1: Summary of the image datasets for multiclass classification,  $R = (d_1 \times d_2)/(d_1 + d_2)$ . “Separate” indicates whether the training data and the testing data are provided separately.

| Dataset      | # train | # test | # features | # classes | $R$   | Separate |
|--------------|---------|--------|------------|-----------|-------|----------|
| Yale_32 × 32 | 110     | 55     | 1024       | 15        | 16    | No       |
| Yale_64 × 64 | 110     | 55     | 4096       | 15        | 32    | No       |
| ORL_32 × 32  | 266     | 134    | 1024       | 40        | 16    | No       |
| ORL_64 × 64  | 266     | 134    | 4096       | 40        | 32    | No       |
| UMist        | 383     | 192    | 10304      | 20        | 50.5  | No       |
| GTface       | 500     | 250    | 307200     | 50        | 274.3 | No       |
| PIE          | 7703    | 3851   | 1024       | 68        | 16    | No       |
| Letter       | 15000   | 5000   | 16         | 26        | 2     | Yes      |
| Shuttle      | 43500   | 14500  | 9          | 7         | 1.5   | Yes      |
| USPS         | 7291    | 2007   | 256        | 10        | 8     | Yes      |
| MNIST        | 60000   | 10000  | 784        | 10        | 14    | Yes      |

Fifty trials have been conducted for each problem when comparing ELM and 2DEM on all datasets. For the face datasets, which do not provide separate training set and testing set, we randomly choose the 2/3 of the total samples as training set and the rest as testing in each trail. For other datasets, we conduct the experiments with fifty different random initializations of ELM and 2DELM. The averaged training accuracy, testing accuracy, training time, and testing time of all trials are recorded, and the comparison of testing accuracy is based on pairwise  $t$ -tests at 95% significance level.

*4.3. Comparison Results.* We first compare the processing time of ELM and 2DELM when dealing with matrix pattern. Since most of the datasets were provided with vectors in Table 1, we find two image databases with samples stored in original pictures. We put the *cell* structure in Matlab which contains matrices as the elements as the input and count the cpu-time for each trial. Note that implementation by vectorization is accelerated by some linear algebra libraries; here we count tics for getting  $\mathbf{H}$  without vectorization in ELM due to the same conditions with 2DELM. The time needed by calculating hidden layer output matrix  $\mathbf{H}$  for ELM or 2DELM is also depend on the number of hidden nodes  $L$ . Figure 3 shows the results when  $L$  is in the range  $\{100, 200, \dots, 1000\}$ . With each  $L$ , mean time and standard deviation of ten trails are shown on the figures. We can see that 2DELM achieves faster speed under the same implementation condition and tends to be more stable with random parameters.

We set the number of hidden nodes  $L = 1000$  in both ELM and 2DELM in the following comparison. Table 2 shows the average training time and testing time. We can see that the training time (time needed for calculating  $\beta$  with  $\mathbf{H}$  by hand) in ELM and 2DELM keeps roughly the same, due to the same size of  $\mathbf{H}$  they share, so as the testing time.

The accuracy comparison results are shown in Table 3. Bold number indicates better mean testing accuracy, and  $\bullet$  indicates this advantage is significant under pairwise  $t$ -tests at 95% significance level ( $\circ$  otherwise). We can see that 2DELM achieves better testing accuracy than ELM in most cases, with the same amounts of hidden nodes and much less input parameters.

TABLE 2: Training time and testing time (in seconds) for ELM and 2DELM, average values on fifty trails.

| Dataset      | Training time |                | Testing time  |               |
|--------------|---------------|----------------|---------------|---------------|
|              | ELM           | 2DELM          | ELM           | 2DELM         |
| Yale_32 × 32 | <b>0.1909</b> | <b>0.1919</b>  | 0.0009        | 0.0022        |
| Yale_64 × 64 | 0.0655        | <b>0.0568</b>  | 0.0053        | <b>0.0003</b> |
| ORL_32 × 32  | 0.1438        | <b>0.1304</b>  | <b>0.0069</b> | 0.0075        |
| ORL_64 × 64  | <b>0.1460</b> | 0.1685         | <b>0.0069</b> | 0.0081        |
| UMist        | <b>0.1987</b> | 0.2290         | 0.0078        | <b>0.0031</b> |
| GTface       | 0.6146        | <b>0.6043</b>  | 0.0197        | <b>0.0122</b> |
| PIE          | <b>6.5056</b> | 6.5642         | <b>0.1532</b> | 0.1554        |
| Letter       | 11.1809       | <b>11.0386</b> | 0.1872        | <b>0.1822</b> |
| Shuttle      | 29.7740       | <b>28.0178</b> | 0.4824        | <b>0.4671</b> |
| USPS         | <b>6.1127</b> | 6.1936         | <b>0.0699</b> | 0.0736        |
| MNIST        | 42.2897       | <b>42.2875</b> | <b>0.3716</b> | 0.3719        |

## 5. Conclusion

In this paper we propose a matrix pattern representation based ELM algorithm 2DELM, which take matrices as input instead of commonly used vectors in the SLFN. The key difference between 2DELM and ELM lies at the feature mapping stage; vectorization is not needed when dealing with matrices and thus reduces the input weights compared with vector pattern case. The learning stage keeps the same as ELM and inherits most characteristics of ELM. The comparing experiments on several image datasets show the effectiveness of the proposed algorithms. In most cases, 2D could achieve better or comparable testing accuracy as ELM while using fewer input weights parameters.

From ELM to 2DELM, we aim to simplify the learning model by reducing parameters while keeping the predicting accuracy under the basic ELM framework. The method also keeps consistent with the general principle of Occam’s razor [21] in classifier design. What is more, for dealing with high dimensional data, the matrix or tensor pattern representation may provide another perspective besides traditional dimension reducing techniques.

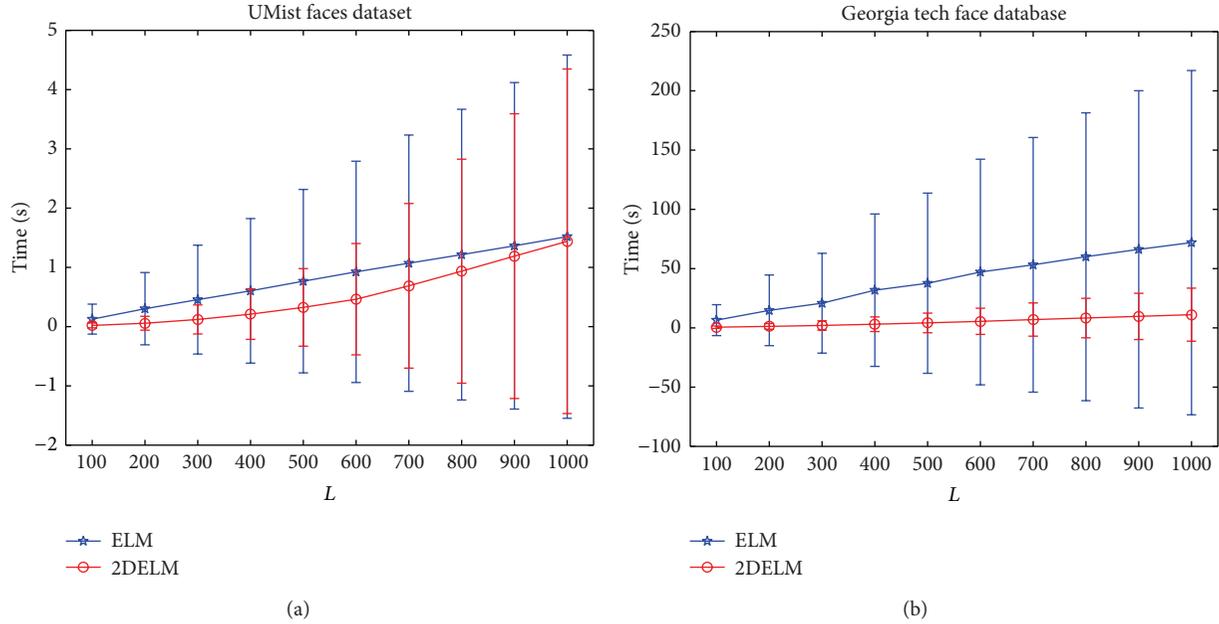


FIGURE 3: The time needed for getting hidden layer output matrix  $H$  on UMist data and Georgia tech face database.

TABLE 3: Mean accuracy ( $\pm$  standard deviation) of fifty-trial comparison of ELM and 2DELm. Testing accuracy is also compared based on pairwise  $t$ -tests at 95% significance level.

| Dataset      | ELM             |                         | 2DELm           |                         |
|--------------|-----------------|-------------------------|-----------------|-------------------------|
|              | Training        | Testing                 | Training        | Testing                 |
| Yale_32 × 32 | 1.0000 ± 0.0000 | 0.7578 ± 0.0585         | 1.0000 ± 0.0000 | <b>0.7880 ± 0.0524*</b> |
| Yale_64 × 64 | 1.0000 ± 0.0000 | 0.8473 ± 0.0454         | 1.0000 ± 0.0000 | <b>0.8720 ± 0.0346*</b> |
| ORL_32 × 32  | 1.0000 ± 0.0000 | 0.9333 ± 0.0233         | 1.0000 ± 0.0000 | <b>0.9416 ± 0.0244*</b> |
| ORL_64 × 64  | 1.0000 ± 0.0000 | 0.9251 ± 0.0258         | 1.0000 ± 0.0000 | <b>0.9378 ± 0.0275*</b> |
| UMist        | 1.0000 ± 0.0000 | 0.9649 ± 0.0139         | 1.0000 ± 0.0000 | <b>0.9749 ± 0.0128*</b> |
| GTface       | 1.0000 ± 0.0000 | <b>0.9752 ± 0.0150°</b> | 1.0000 ± 0.0000 | 0.9746 ± 0.0151         |
| PIE          | 0.9766 ± 0.0016 | 0.9314 ± 0.0047         | 0.9888 ± 0.0009 | <b>0.9586 ± 0.0028*</b> |
| Letter       | 0.9543 ± 0.0011 | <b>0.9366 ± 0.0020*</b> | 0.9489 ± 0.0009 | 0.9311 ± 0.0018         |
| Shuttle      | 0.9995 ± 0.0000 | <b>0.9977 ± 0.0001*</b> | 0.9989 ± 0.0001 | 0.9970 ± 0.0001         |
| USPS         | 0.9910 ± 0.0006 | 0.9369 ± 0.0027         | 0.9918 ± 0.0007 | <b>0.9395 ± 0.0033*</b> |
| MNIST        | 0.9378 ± 0.0010 | 0.9357 ± 0.0016         | 0.9407 ± 0.0010 | <b>0.9403 ± 0.0016*</b> |

• or ° indicates whether the advantage is significant or not.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgments

This research was supported by the National Technology Research and Development Program of China (863 Program) 2012AA01A510. The authors would also like to thank the anonymous reviewers for their patient work and suggestions to improve the paper.

### References

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] H. Kong, L. Wang, E. K. Teoh, X. Li, J.-G. Wang, and R. Venkateswarlu, "Generalized 2D principal component analysis for face image representation and recognition," *Neural Networks*, vol. 18, no. 5-6, pp. 585–594, 2005.

[3] Z. Wang, S. Chen, J. Liu, and D. Zhang, "Pattern representation in feature extraction and classifier design: matrix versus vector," *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 758–769, 2008.

[4] Z. Wang and S. Chen, "New least squares support vector machines based on matrix patterns," *Neural Processing Letters*, vol. 26, no. 1, pp. 41–56, 2007.

[5] J. Yang, D. Zhang, A. F. Frangi, and J.-Y. Yang, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

- [6] J. Ye, R. Janardan, Q. Li et al., "Two-dimensional linear discriminant analysis," in *Proceedings of the Neural Information Processing Systems Conference (NIPS '04)*, vol. 4, p. 4, 2004.
- [7] D. Tao, X. Li, X. Wu, W. Hu, and S. J. Maybank, "Supervised tensor learning," *Knowledge and Information Systems*, vol. 13, no. 1, pp. 1–42, 2007.
- [8] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [9] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [10] Q. Liu, Q. He, and Z. Shi, "Extreme support vector machine classifier," in *Advances in Knowledge Discovery and Data Mining*, pp. 222–233, Springer, New York, NY, USA, 2008.
- [11] B. Frénay and M. Verleysen, "Using SVMs with randomised feature spaces: an extreme learning approach," in *Proceedings of the 18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN '10)*, pp. 315–320, April 2010.
- [12] K. Choi, K.-A. Toh, and H. Byun, "Incremental face recognition for large-scale social network services," *Pattern Recognition*, vol. 45, no. 8, pp. 2868–2883, 2012.
- [13] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on bow framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, 2014.
- [14] D. Li, Z. Pan, Z. Deng, and Y. Zhang, "Large scale extreme learning machine using MapReduce," *International Journal of Digital Content Technology and Its Applications*, vol. 6, no. 20, pp. 62–70, 2012.
- [15] Y. Wang, D. Li, Y. Du, and Z. Pan, "Anomaly detection in traffic using L1-norm minimization extreme learning machine," *Neurocomputing*, vol. 149, pp. 415–425, 2015.
- [16] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- [17] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pp. 389–395, April 2009.
- [18] A. Castaño, F. Fernández-Navarro, and C. Hervás-Martínez, "PCA-ELM: a robust and pruned extreme learning machine approach based on principal component analysis," *Neural Processing Letters*, vol. 37, no. 3, pp. 377–392, 2013.
- [19] D. B. Graham and N. M. Allinson, "Face recognition: from theory to applications," *NATO ASI Series F: Computer and Systems Sciences*, vol. 163, pp. 446–456, 1998.
- [20] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615–1618, 2003.
- [21] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, New York, NY, USA, 2012.

## Research Article

# Extreme Learning Machine Assisted Adaptive Control of a Quadrotor Helicopter

Yu Zhang,<sup>1</sup> Zheng Fang,<sup>2</sup> and Hongbo Li<sup>3</sup>

<sup>1</sup>School of Aeronautics and Astronautics, Zhejiang University, Zhejiang, Hangzhou 310027, China

<sup>2</sup>State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Liaoning, Shenyang 110189, China

<sup>3</sup>Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Zheng Fang; [fangzheng@mail.neu.edu.cn](mailto:fangzheng@mail.neu.edu.cn)

Received 21 August 2014; Revised 10 December 2014; Accepted 14 December 2014

Academic Editor: Yi Jin

Copyright © 2015 Yu Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Control of quadrotor helicopters is difficult because the problem is naturally nonlinear. The problem becomes more challenging for common model based controllers when unpredictable uncertainties and disturbances in physical control system are taken into account. This paper proposes a novel intelligent controller design based on a fast online learning method called extreme learning machine (ELM). Our neural controller does not require precise system modeling or prior knowledge of disturbances and well approximates the dynamics of the quadrotor at a fast speed. The proposed method also incorporates a sliding mode controller for further elimination of external disturbances. Simulation results demonstrate that the proposed controller can reliably stabilize a quadrotor helicopter in both agitated attitude and position control tasks.

## 1. Introduction

Unmanned aerial vehicles (UAVs) have received considerable attention in recent years due to their wide military and civilian applications. Their typical applications include collecting data, monitoring, surveillance, investigation, and inspection [1], which can be used in scenarios such as environmental monitoring, resource exploration, agriculture surveying, traffic control, weather forecasting, aerial photography, disasters search, and rescue. Particularly, unmanned rotorcrafts play an important role in these applications because of their flexibility such as hovering and vertical take-off and landing (VTOL). However, conventional rotorcraft with a main rotor and a tail rotor has extremely complex dynamics. Its maneuverability is greatly limited since it is very difficult to design a controller with high performance. Meanwhile, a special kind of rotorcrafts called quadrotor helicopter which has a compact form is becoming more and more popular than conventional rotorcrafts as they are mechanically and dynamically simpler and easier to control. In spite of this, the quadrotor is still a dynamically unstable system and its controller design is also challenging because of the inherent system characteristics

such as nonlinearities, cross couplings due to the gyroscopic effects, and underactuation [2]. Besides, all the applications mentioned above require a vehicle with stable and accurate performance of motion control. Therefore, how to design a high quality controller for quadrotor is an important and meaningful problem.

Many different control theories and methods are employed to design the attitude stabilizer or motion controller for quadrotors [3]. In the early stage, Bouabdallah et al. first apply two different control techniques, PID and linear quadratic (LQ) techniques, to a microquadrotor called OS4 [2]. Subsequently, other research works using PID [4, 5] or LQ [6] methods are also reported. These two kinds of controllers are easy to design and implement. However, they cannot handle the unmolded dynamics and external disturbances. Because of the nonlinearity, feedback linearization technology is adopted to design the controller for quadrotors [7, 8]. However, precise model of quadrotors which is required for linearization is difficult to obtain. Backstepping design method [9–11] is another choice to deal with the nonlinear models of quadrotors [12, 13], but the controller is usually vulnerable to parameters uncertainty. To deal with the dynamic

uncertainty and external disturbances, sliding mode control [14] and  $H$  infinite control [15, 16] algorithms are used to improve the robustness of the system. However, sliding mode controller is likely to have chattering phenomena in both sides of sliding mode surface due to the delay of sensors or actuators, and  $H$  infinite control method, which requires approximate linearization near the equilibrium point of the system, is not suitable for aggressive control.

The performance of model based controllers above degrades significantly in case of model uncertainties and unknown disturbances. One potential way to solve this problem is intelligent control methods such as fuzzy logic control [17, 18], neural networks control [19, 20], and learning based control [21]. One main challenge of utilizing these techniques is the convergence performance of controllers. Slow convergence may cause failure in real time control system.

In this paper, a novel computational intelligence technique called extreme learning machine (ELM) [22] is introduced to control the quadrotors by compensating the dynamic uncertainties and the external disturbances. ELM theories have been successfully improved recently by Cao et al. [23] and widely used in several control systems [24]. Essentially, it is a learning policy for generalized single hidden layer feedforward networks (SLFNs) whose input weight and hidden layer do not need to be tuned. Compared with backpropagation (BP) method and support vector machines (SVMs), ELM provides better generalization performance at a much faster learning speed and with least human intervention [25]. Thus, ELM can be used to estimate and compensate the uncertainties and disturbances of the systems simultaneously in real time.

There are three main contributions of this paper. First, we employ Lyapunov second method to minimize the cost function of ELM and satisfy the quadrotor control system stability simultaneously under the framework of ELM. So it is a development of ELM theory. Second, traditional neural network based controller is facing two problems. One is that too many parameters need to be initialized and tuned. The other is slow convergence. Since ELM can converge very fast with its input weight and hidden nodes parameters fixed, the two problems above are significantly alleviated when ELM is employed to the quadrotor control. Third, the quadrotor control system is a complicated dynamic system. The stability of the ELM based control system is proved.

This paper is organized as follows: In Section 2, the mathematical model of ELM is presented. Kinematics and dynamics models of quadrotors are described in Section 3. In Section 4, the details of designing an ELM-assisted quadrotor controller are presented. The stability of the proposed controller is also proved in this section. Simulation results are given in Section 5 to demonstrate the performance of the proposed controller. Finally, the paper is concluded in Section 6.

## 2. Preliminary on Extreme Learning Machine

In this section, the basic idea of ELM is briefly reviewed to provide a background for designing controller for the quadrotor. ELM is a special SLFN whose learning speed

can be much faster than conventional feedforward network learning algorithm such as BP algorithm while obtaining better generalization performance [26]. The essence of ELM is that the input weights and the parameters of the hidden layer do not need to adjust during the learning procedure. We take a SLFN with  $L$  hidden nodes as an example. The output of the SLFN can be modeled as

$$f_L(\mathbf{x}) = \sum_{i=1}^L \beta_i G(\mathbf{x}, \mathbf{c}_i, a_i), \quad \mathbf{x} \in \mathbf{R}^n, \quad \mathbf{c}_i \in \mathbf{R}^n, \quad (1)$$

where  $\beta_i$  is the output weight connecting the  $i$ th hidden node to the output node,  $G(\mathbf{x}, \mathbf{c}_i, a_i)$  is the activation function of the  $i$ th hidden node, and  $\mathbf{c}_i$  and  $a_i$  are the parameters of the activation function which are randomly generated and then fixed afterwards. Furthermore, there are two kinds of hidden nodes. Usually, additive hidden nodes use Sigmoid or threshold activation function as follows:

$$G(\mathbf{x}, \mathbf{c}_i, a_i) = \frac{1 - e^{-(\mathbf{c}_i \mathbf{x} + a_i)}}{1 + e^{-(\mathbf{c}_i \mathbf{x} + a_i)}}, \quad (2)$$

where  $\mathbf{c}_i$  is the input weight vector for the  $i$ th hidden node and  $a_i$  is the bias of the  $i$ th hidden node. For RBF hidden nodes, Gaussian or triangular activation function is used for activation which can be given by

$$G(\mathbf{x}, \mathbf{c}_i, a_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2a_i^2}\right), \quad (3)$$

where  $\mathbf{c}_i$  and  $a_i$  are the center and impact factor of the  $i$ th RBF node, respectively.

Then,  $N$  sample pairs  $(\mathbf{x}_k, \mathbf{y}_k) \in \mathbf{R}^n \times \mathbf{R}^m$  ( $k = 1, \dots, N$ ) are used to train the SLFN. If this network can approximate  $N$  samples with zero error, there must exist  $\beta_i^*$ ,  $\mathbf{c}_i$ , and  $a_i$  such that

$$\sum_{i=1}^L \beta_i^* G(\mathbf{x}_k, \mathbf{c}_i, a_i) = \mathbf{y}_k, \quad k = 1, \dots, N. \quad (4)$$

The previous equation can be rewritten compactly as

$$\mathbf{H}\boldsymbol{\beta}^* = \mathbf{Y}, \quad (5)$$

where

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_1, \mathbf{c}_L, a_L) \\ \vdots & \ddots & \vdots \\ G(\mathbf{x}_N, \mathbf{c}_1, a_1) & \cdots & G(\mathbf{x}_N, \mathbf{c}_L, a_L) \end{bmatrix}_{N \times L}, \quad (6)$$

$$\boldsymbol{\beta}^* = [\beta_1^{*T} \cdots \beta_L^{*T}]^T,$$

$$\mathbf{Y} = [\mathbf{Y}_1^T \cdots \mathbf{Y}_L^T]^T.$$

ELM aims to minimize not only the training error but also the norm of output weights, which would yield a better generalization performance [25]. In other words, ELM try to minimize the training error as well as the norm of the output weights. So the objective function can be expressed as

$$\min \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|, \|\boldsymbol{\beta}\|. \quad (7)$$

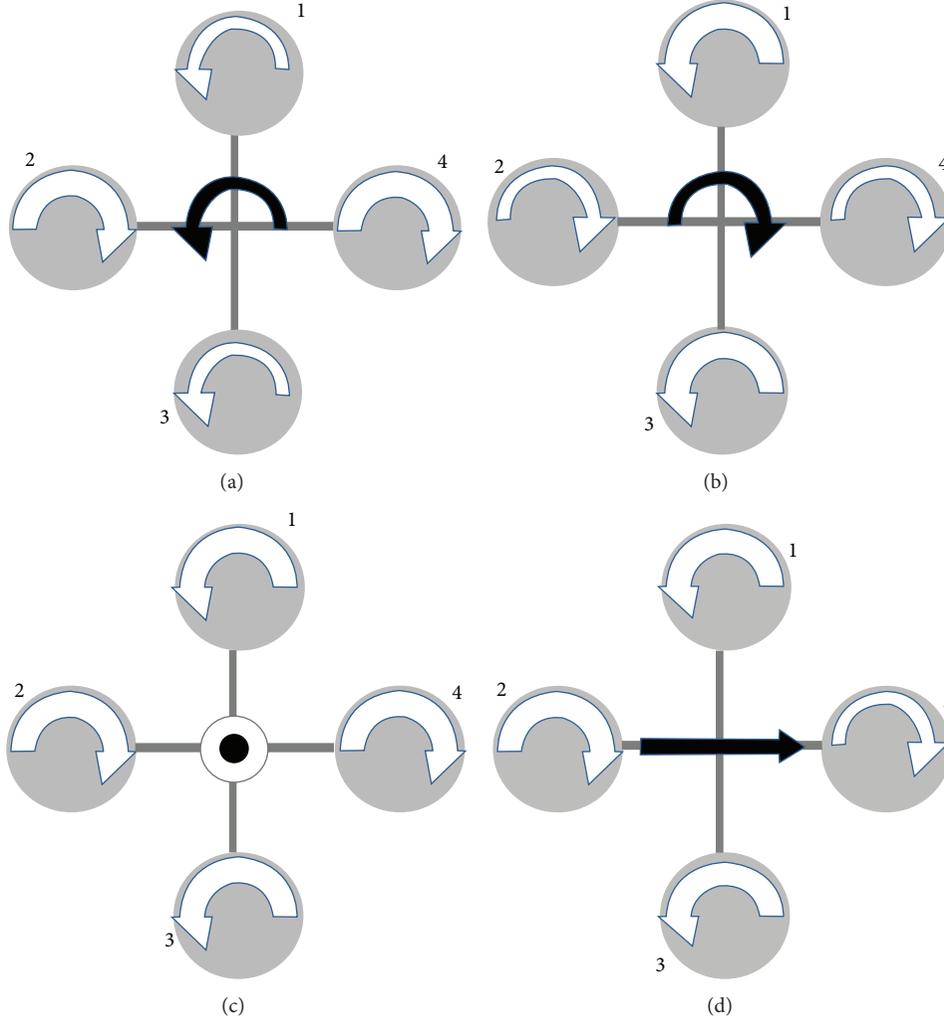


FIGURE 1: Quadrotor schematic. The white arrow width is proportional to rotor rotational speed, and the black arrows show the moving direction of the quadrotor.

Finally, the minimal norm least-square method instead of the standard optimization method was adopted in the original implementation of ELM [25], and the closed form solution is obtained:

$$\boldsymbol{\beta}^* = \mathbf{H}^\dagger \mathbf{Y}, \quad (8)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ .

### 3. Quadrotor Helicopters Model

The quadrotor helicopter has four rotors in cross configuration. As we can see from Figure 1, the two pairs of rotors (1, 3) and (2, 4) always turn in opposite directions. By changing the rotor speed, we can move the vehicles in different directions in 3D space. Initially, suppose all the rotors have the same speed as shown in Figure 1(c); increasing the four rotors speeds together generates upward movement. Then, increasing or decreasing 2 and 4 rotors speed inversely

will change the attitude of the quadrotor. It generates roll rotation as well as lateral motion. Changing 1 and 3 rotors speed in the same way produces the pitch rotation as well as the longitudinal movements (see Figure 1(d)). Finally, if the counter-torque resulting from rotor (1, 3) is different from that of rotor (2, 4), the yaw rotations of the quadrotor are generated as shown in Figures 1(a) and 1(b).

*3.1. Kinematic Model.* To facilitate the model description, quadrotor reference frames are defined first. We consider earth fixed frame  $E\text{-}XYZ$  as an inertia frame while frame  $B\text{-}xyz$  is a body fixed frame as shown in Figure 2.

To transform an attitude from the body frame ( $x$ ,  $y$ , and  $z$ ) to the inertia frame ( $X$ ,  $Y$ , and  $Z$ ), the coordinate transformation matrix [1]

$$\mathbf{R} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \quad (9)$$

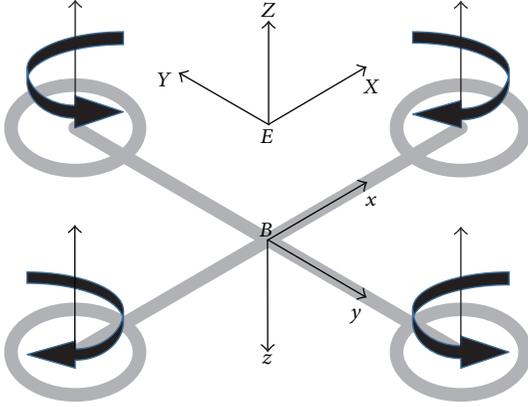


FIGURE 2: Quadrotor reference frames.

where  $c$  and  $s$  denote cosine and sine functions. Then, the relationship between the euler rates  $\dot{\eta} = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$  and angular body rates  $\omega = [p \ q \ r]^T$  can be represented by

$$\omega = \mathbf{R}_r \dot{\eta}, \quad (10)$$

where

$$\mathbf{R}_r = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi) \cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi) \cos(\theta) \end{bmatrix}. \quad (11)$$

Particularly, when the quadrotor works around the hover position, the following condition can be approximately satisfied:

$$\mathbf{R}_r \approx \mathbf{I}_{3 \times 3}, \quad (12)$$

where  $\mathbf{I}$  is the identity matrix.

**3.2. Dynamic Model.** Since the quadrotor helicopter can fly in 3D space, both rotational and translational motions should be considered in system modeling. The rotational equations of motion can be derived in the body frame using the Newton-Euler method while the translational equations of motion are derived in the inertia navigation frame using Newton's second law [1]. Many works about quadrotor modeling have been reported, so here we directly give the full state space model [27] of a quadrotor for controller design.

The state vector is defined as  $\mathbf{X} = [\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ z \ \dot{z} \ x \ \dot{x} \ y \ \dot{y}]^T$ , where  $\phi$ ,  $\theta$ , and  $\psi$  are the attitude angles which represent roll, pitch, and yaw.  $(x, y, z)$  is the quadrotor's position in the inertia frame. If we define the speeds of the four rotors as  $\Omega_1$ ,  $\Omega_2$ ,  $\Omega_3$ , and  $\Omega_4$ , the control input vector can be further defined as  $\mathbf{U} = [U_1 \ U_2 \ U_3 \ U_4]^T$ , which is mapped by

$$\begin{aligned} U_1 &= k_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \\ U_2 &= k_F(-\Omega_2^2 + \Omega_4^2), \end{aligned}$$

$$\begin{aligned} U_3 &= k_F(\Omega_1^2 - \Omega_3^2), \\ U_4 &= k_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2), \end{aligned} \quad (13)$$

where  $k_F$  and  $k_M$  are the aerodynamic force and moment constants, respectively. In this case,  $U_1$  is the total thrust generated from the four rotors.  $U_2$ ,  $U_3$ , and  $U_4$  are the equivalent of the torques around  $x$ -,  $y$ -, and  $z$ -axis in body frame.

Then, the state space model can be given by the following compact form:

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, \mathbf{U}), \quad (14)$$

where

$$\mathbf{f}(\mathbf{X}, \mathbf{U}) = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\psi a_1 + \dot{\theta}a_2\Omega_r + b_1U_2 \\ \dot{\theta} \\ \dot{\phi}\psi a_3 - \dot{\phi}a_4\Omega_r + b_2U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}a_5 + b_3U_4 \\ \dot{z} \\ g - \frac{(\cos\phi \cos\theta)U_1}{m} \\ \dot{x} \\ \frac{(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)U_1}{m} \\ \dot{y} \\ \frac{(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)U_1}{m} \end{pmatrix}, \quad (15)$$

$$a_1 = \frac{(I_{yy} - I_{zz})}{I_{xx}},$$

$$a_2 = \frac{J_r}{I_{xx}},$$

$$a_3 = \frac{(I_{zz} - I_{xx})}{I_{yy}},$$

$$a_4 = \frac{J_r}{I_{yy}},$$

$$a_5 = \frac{(I_{xx} - I_{yy})}{I_{zz}},$$

$$b_1 = \frac{l}{I_{xx}},$$

$$b_2 = \frac{l}{I_{yy}},$$

$$b_3 = \frac{1}{I_{zz}},$$

and  $I_{xx}$ ,  $I_{yy}$ , and  $I_{zz}$  are the moment of inertia around  $x$ -,  $y$ -, and  $z$ -axis, respectively.  $J_r$  is the propeller inertia coefficient.  $l$  is the arm length of the quadrotor.

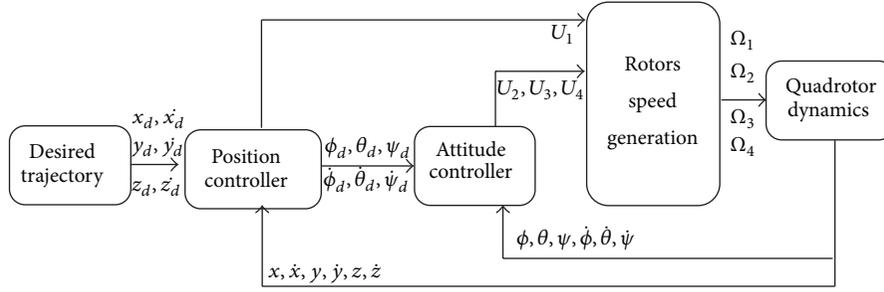


FIGURE 3: The structure of the quadrotor control system.

#### 4. Controller Design Using ELM

By investigating the relationship between the state variables in (15), the attitude of the quadrotor does not depend on the translational motion while the translation of the quadrotor depends on the attitude angles. Thus, the whole system can be decoupled into two subsystems: inner loop attitude subsystem and outer loop position subsystem, respectively. The structure of the whole control system is described in Figure 3, where  $(x_d, \dot{x}_d, y_d, \dot{y}_d, z_d, \dot{z}_d)$  is the coordinate of the desired trajectory and  $(\phi_d, \theta_d, \psi_d, \dot{\phi}_d, \dot{\theta}_d, \dot{\psi}_d)$  is the desired attitude angles and their rates generated by the position controller.  $U_1$  is produced by the position controller related to the altitude of the vehicle while the  $U_2, U_3,$  and  $U_4$  are calculated by the attitude controller.

**4.1. Position Control Loop.** The outer loop is a position control system which is much slower compared to the inner loop attitude control system. In addition, the desired roll, pitch, and yaw angles are normally small that makes the position subsystem an approximately linear system near the equilibrium points. A simple PD controller is sufficient to control the quadrotor's position.

For horizontal movement, according to the desired trajectory, the expected accelerations in  $X$  and  $Y$  direction can be calculated by the PD controller:

$$\begin{aligned}\ddot{x}_d &= k_{px}(x_d - x) + k_{dx}(\dot{x}_d - \dot{x}), \\ \ddot{y}_d &= k_{py}(y_d - y) + k_{dy}(\dot{y}_d - \dot{y}),\end{aligned}\quad (17)$$

where  $(k_{px}, k_{dx})$  and  $(k_{py}, k_{dy})$  are the proportional and differential control gains in  $X$  and  $Y$  directions. Control gains are acquired using pole placement design method to make the position system stable. When the yaw angle of the quadrotor keeps fixed, we can substitute  $\ddot{x}_d$  and  $\ddot{y}_d$  into (15) and obtain

$$\begin{aligned}\frac{(\cos \phi_d \sin \theta_d \cos \psi + \sin \phi_d \sin \psi)U_1}{m} &= \ddot{x}_d, \\ \frac{(\cos \phi_d \sin \theta_d \sin \psi - \sin \phi_d \cos \psi)U_1}{m} &= \ddot{y}_d.\end{aligned}\quad (18)$$

Using the small angle assumption around the equilibrium position, (18) can be simplified as

$$\begin{aligned}\frac{(\theta_d \cos \psi + \phi_d \sin \psi)U_1}{m} &= \ddot{x}_d, \\ \frac{(\theta_d \sin \psi - \phi_d \cos \psi)U_1}{m} &= \ddot{y}_d.\end{aligned}\quad (19)$$

Then, the reference roll and pitch angles can be solved by

$$\begin{bmatrix} \phi_d \\ \theta_d \end{bmatrix} = \frac{m}{U_1} \begin{bmatrix} \sin \psi & \cos \psi \\ -\cos \psi & \sin \psi \end{bmatrix}^{-1} \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \end{bmatrix}.\quad (20)$$

For vertical movements, the altitude control input can be calculated by

$$\begin{aligned}U_1 &= \frac{1}{\cos(\phi) \cos(\theta)} \\ &\cdot ((1 + k_1 k_2)(z_d - z) + (k_1 + k_2)(\dot{z}_d - \dot{z}) + mg)\end{aligned}\quad (21)$$

using backstepping design method, where  $k_1$  and  $k_2$  are positive real numbers and  $mg$  is the term to balance the gravity. Please note that all the inputs have their own saturation functions.

**4.2. Attitude Control Loop.** The quadrotor's attitude system is sensitive to the disturbances. It is also a nonlinear system with unmodeled dynamic uncertainties compared to the real quadrotor system. Hence an ELM based SLFN with very fast learning speed is employed to adaptively learn and estimate the nonlinear model including the uncertainties of the quadrotor's attitude system in real time. Meanwhile, the proposed neural controller incorporates a sliding mode controller to deal with the external disturbances. The stability of the attitude control system is proved using the Lyapunov theory.

Since the roll, pitch, and yaw subsystems have the same form of expression, without loss of generality, we take roll subsystem as an example to introduce the design procedure. Although three attitude subsystems are coupled with each other, the coupled items can be considered the unknown nonlinear functions of dynamic system or internal disturbances and thus be adaptively estimated by the SLFN with ELM.

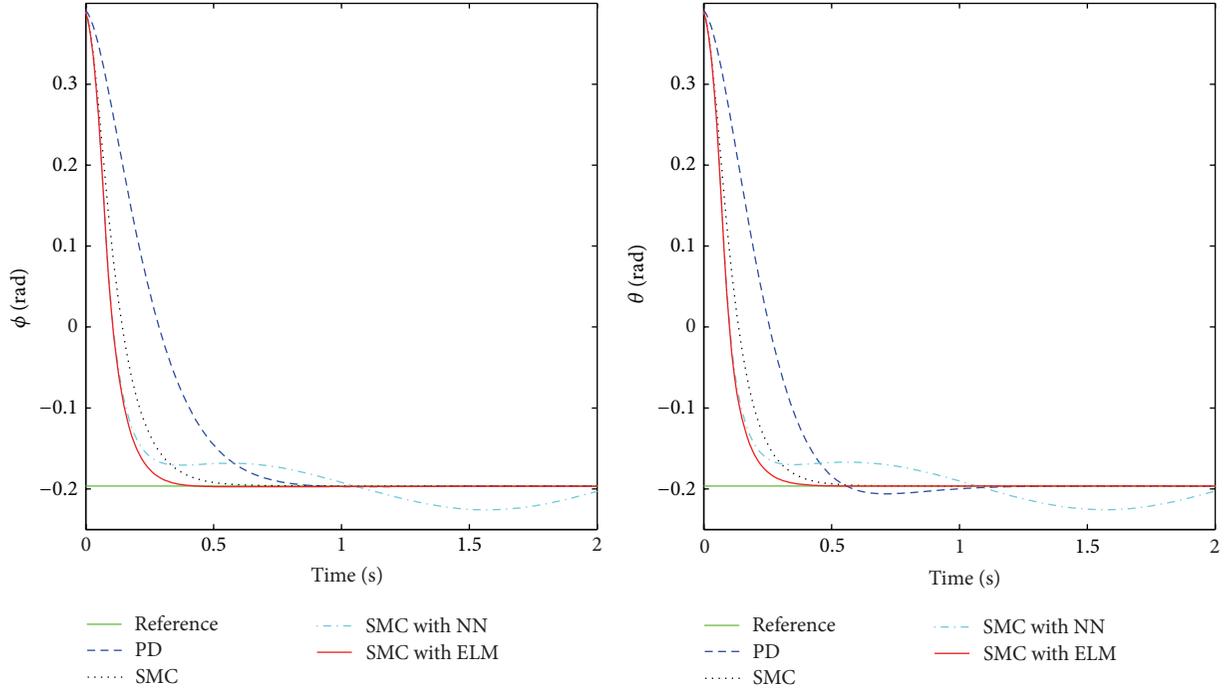


FIGURE 4: Roll and pitch angle regulation.

4.2.1. *Model Based Control Law.* According to (15), the roll subsystem is expressed in the following form:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= f(\mathbf{x}) + g(\mathbf{x})u + d, \\ y &= x_1, \end{aligned} \quad (22)$$

where  $f(\mathbf{x})$  is assumed to be unknown but bounded and  $g(\mathbf{x})$  is the input gain. Particularly, in roll subsystem  $g(\mathbf{x}) = b_1 = l/I_{xx}$  which is known.  $d$  is the unknown disturbances. It is bounded and satisfies  $|d| \leq d_{\max}$ , where  $d_{\max}$  is the upper bound of the disturbances. Then, the output tracking error  $\mathbf{E} = [e, \dot{e}]^T$ , where  $e = y_d - x_1$  and  $\dot{e} = \dot{y}_d - x_2$ .

Suppose all the functions and parameters in (22) are precisely known; the perfect control law  $u^*$  can be obtained using feedback linearization method as follows:

$$u^* = \frac{I_{xx}}{l} (\ddot{y}_d - f(\mathbf{x}) - d + \mathbf{K}^T \mathbf{E}), \quad (23)$$

where  $\mathbf{K} = [K_1, K_2]^T$  is a real number vector. Substitute (23) into (22); the following error equation is obtained:

$$\ddot{e} + K_2 \dot{e} + K_1 e = 0. \quad (24)$$

$K_1$  and  $K_2$  can be determined when all the roots of the polynomial  $s^2 + K_2 s + K_1 = 0$  are in the open left half plane, which indicates that the tracking error will converge to zero.

4.2.2. *Neural Controller Structure.* Since  $f(\mathbf{x})$  is coupled with other variables and cannot be accurately modeled,  $d$

is also unknown disturbances; control law of (23) cannot be directly implemented. Here, we present an ELM-assisted neural controller to solve the problem.

The inner loop controller is mainly composed of two parts. One is the neural controller  $u_n$  and the other is the sliding mode controller  $u_s$ . So the overall control law

$$u = u_n + u_s. \quad (25)$$

The neural controller is used to approximate the unknown function  $f(\mathbf{x})$  while the sliding mode controller is employed to eliminate the external disturbances  $d$ .

For the neural controller, according to (23), the optimal neural control law is expected as

$$u_n^* = \frac{I_{xx}}{l} (\ddot{y}_d - f(\mathbf{x}) + \mathbf{K}^T \mathbf{E}). \quad (26)$$

Here, a SLFN whose parameters are determined based on the ELM is employed to approximate the above desired neural control law. Then, the actual neural control law  $u_n$  is given by

$$u_n = \mathbf{H}(\mathbf{r}, \mathbf{c}, \mathbf{a}) \boldsymbol{\beta}, \quad (27)$$

where  $\mathbf{r} = [\ddot{y}_d; \mathbf{x}; \mathbf{E}]^T$  is the input vector. As we mentioned in Section 2,  $\mathbf{c}$  and  $\mathbf{a}$  are hidden node parameters which are generated randomly and then fixed. Training a SLFN with ELM is the equivalent of finding a least-square solution of the output weights  $\boldsymbol{\beta}^*$ , such that

$$\hat{u}_n^* = \mathbf{H}(\mathbf{r}, \mathbf{c}, \mathbf{a}) \boldsymbol{\beta}^* + \boldsymbol{\varepsilon}(\mathbf{r}), \quad (28)$$

where  $\boldsymbol{\varepsilon}(\mathbf{r})$  is the approximation error. This error arises if the number of hidden nodes is much less than the number of

training samples [24], but it is bounded with the constant  $\varepsilon_N$ ; that is,  $|\varepsilon(\mathbf{r})| \leq \varepsilon_N$ .

For the sliding mode controller, standard sliding mode control law has the form

$$u_s = G_s(\mathbf{x}) \operatorname{sgn}(f(\mathbf{E})), \quad (29)$$

where  $G_s(\mathbf{x})$  is the sliding mode gain and  $f(\mathbf{E})$  is the sliding mode surface function.

Since we have the forms of the neural controller and the sliding mode controller, the next step is to derive the adaptive law  $\dot{\boldsymbol{\beta}}$ , so that  $\boldsymbol{\beta}$  converges to  $\boldsymbol{\beta}^*$ . The parameters of the sliding mode controller are determined. The details are given in the next subsection.

**4.3. Adaptive Control Law and Stability Analysis.** The stable adaptive law for  $\boldsymbol{\beta}$  and the parameters of the sliding mode controller are derived using Lyapunov second method.

Based on (22), (25), and (26), the tracking error equation can be obtained:

$$\dot{\mathbf{E}} = \boldsymbol{\Lambda}\mathbf{E} + \mathbf{B} \left[ u_n^* - u_n - u_s - \frac{dI_{xx}}{l} \right], \quad (30)$$

where

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & 1 \\ -K_1 & -K_2 \end{bmatrix}, \quad (31)$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ l \\ I_{xx} \end{bmatrix}.$$

Substituting (27), (28) into (30) yields

$$\dot{\mathbf{E}} = \boldsymbol{\Lambda}\mathbf{E} + \mathbf{B} \left[ \mathbf{H}(\mathbf{r}, \mathbf{c}, \mathbf{a})\tilde{\boldsymbol{\beta}} + \varepsilon(\mathbf{r}) - u_s - \frac{dI_{xx}}{l} \right], \quad (32)$$

where  $\tilde{\boldsymbol{\beta}} = \boldsymbol{\beta}^* - \boldsymbol{\beta}$ . Then, the following Lyapunov function is considered to derive the stable tuning law for  $\boldsymbol{\beta}$ :

$$V = \frac{1}{2}\mathbf{E}^T\mathbf{P}\mathbf{E} + \frac{1}{2\eta}\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}}, \quad (33)$$

where  $\mathbf{P}$  is a symmetric and positive definite matrix and  $\eta$  is a positive constant which is referred to as the learning rate of the SLFN.

According to (32), the derivative of the Lyapunov function is acquired as

$$\begin{aligned} \dot{V} &= \frac{1}{2} \left[ \dot{\mathbf{E}}^T\mathbf{P}\mathbf{E} + \mathbf{E}^T\mathbf{P}\dot{\mathbf{E}} \right] + \frac{1}{\eta}\tilde{\boldsymbol{\beta}}^T\dot{\tilde{\boldsymbol{\beta}}} \\ &= -\frac{1}{2}\mathbf{E}^T\mathbf{Q}\mathbf{E} + \left( \mathbf{E}^T\mathbf{P}\mathbf{B}\mathbf{H} + \frac{1}{\eta}\tilde{\boldsymbol{\beta}}^T \right) \tilde{\boldsymbol{\beta}} \\ &\quad - \mathbf{E}^T\mathbf{P}\mathbf{B} \left( \frac{dI_{xx}}{l} - \varepsilon(\mathbf{r}) \right) - \mathbf{E}^T\mathbf{P}\mathbf{B}u_s, \end{aligned} \quad (34)$$

where

$$\mathbf{Q} = -(\boldsymbol{\Lambda}^T\mathbf{P} + \mathbf{P}\boldsymbol{\Lambda}). \quad (35)$$

Suppose  $\mathbf{Q}$  is selected as a symmetric definite matrix, the first item of the Lyapunov function will converge to zero. Then,  $\mathbf{P}$  can be calculated by solving (35). According to the second item of the Lyapunov function,  $\tilde{\boldsymbol{\beta}}$  can be eliminated if we let

$$\dot{\tilde{\boldsymbol{\beta}}}^T = -\eta\mathbf{E}^T\mathbf{P}\mathbf{B}\mathbf{H}. \quad (36)$$

Consider  $\dot{\tilde{\boldsymbol{\beta}}} = -\dot{\tilde{\boldsymbol{\beta}}}$  based on the definition of the  $\tilde{\boldsymbol{\beta}}$ . Thus, the tuning rule for the output weight is acquired:

$$\dot{\tilde{\boldsymbol{\beta}}}^T = \eta\mathbf{E}^T\mathbf{P}\mathbf{B}\mathbf{H}. \quad (37)$$

After that, (34) becomes

$$\dot{V} = -\frac{1}{2}\mathbf{E}^T\mathbf{Q}\mathbf{E} - \mathbf{E}^T\mathbf{P}\mathbf{B} \left( \frac{dI_{xx}}{l} - \varepsilon(\mathbf{r}) \right) - \mathbf{E}^T\mathbf{P}\mathbf{B}u_s. \quad (38)$$

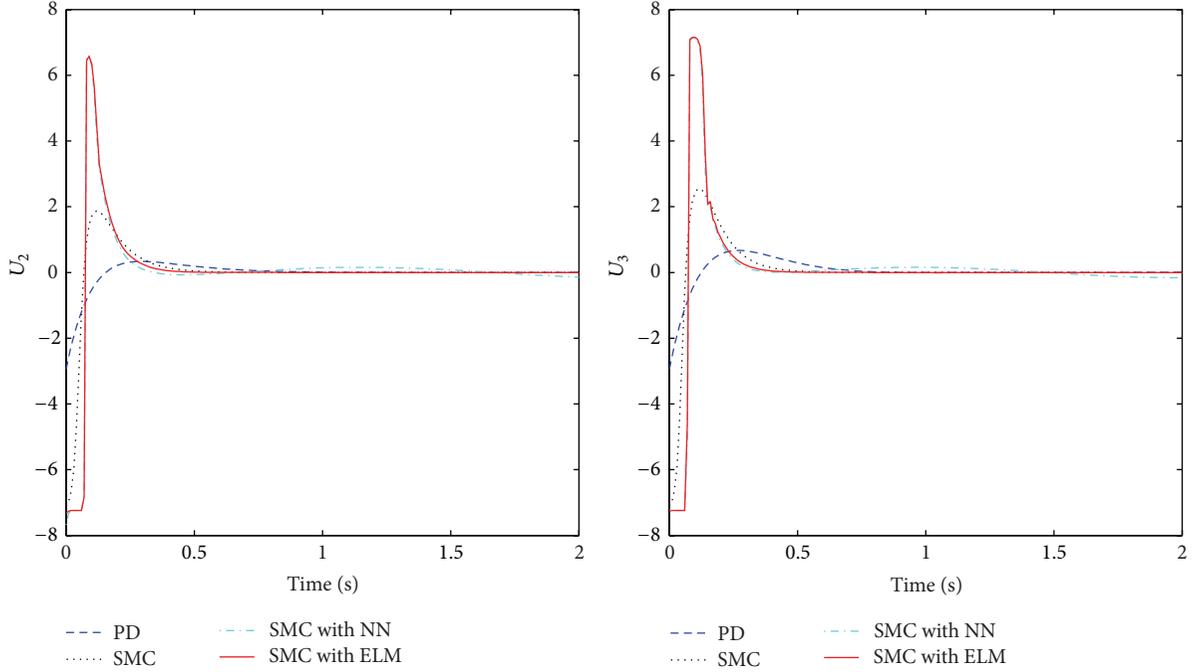
To make (38) less than or equal to zero, the sliding mode controller  $u_s$  can be determined as

$$u_s = \left( \frac{d_{\max}I_{xx}}{l} + \varepsilon_N \right) \operatorname{sgn}(\mathbf{E}^T\mathbf{P}\mathbf{B}). \quad (39)$$

Therefore, the following relationship can be derived from (39):

$$\begin{aligned} \dot{V} &\leq -\frac{1}{2}\mathbf{E}^T\mathbf{Q}\mathbf{E} + |\mathbf{E}^T\mathbf{P}\mathbf{B}| \left( \frac{|dI_{xx}}{l} + |\varepsilon(\mathbf{r})| \right) \\ &\quad - \mathbf{E}^T\mathbf{P}\mathbf{B} \left( \left( \frac{d_{\max}I_{xx}}{l} + \varepsilon_N \right) \operatorname{sgn}(\mathbf{E}^T\mathbf{P}\mathbf{B}) \right) \\ &= -\frac{1}{2}\mathbf{E}^T\mathbf{Q}\mathbf{E} + |\mathbf{E}^T\mathbf{P}\mathbf{B}| \left( \frac{|dI_{xx}}{l} + |\varepsilon(\mathbf{r})| \right) \\ &\quad - |\mathbf{E}^T\mathbf{P}\mathbf{B}| \left( \frac{d_{\max}I_{xx}}{l} + \varepsilon_N \right) \\ &= -\frac{1}{2}\mathbf{E}^T\mathbf{Q}\mathbf{E} + |\mathbf{E}^T\mathbf{P}\mathbf{B}| (|d| - d_{\max}) \frac{I_{xx}}{l} \\ &\quad + |\mathbf{E}^T\mathbf{P}\mathbf{B}| (|\varepsilon(\mathbf{r})| - \varepsilon_N) \\ &\leq 0. \end{aligned} \quad (40)$$

Equation (40) indicates that  $\dot{V}$  is negative semidefinite and the control system is guaranteed to be stable. Please note that traditional ELM trains their output weights using the least-square error method, but in this paper, we derived the tuning rate of the output weights using Lyapunov second method. It is an online sequential learning process that makes the controller applicable in real time. The input weight and the parameters of the hidden nodes are randomly assigned as the normal ELM algorithm.

FIGURE 5: Control inputs  $U_2$  and  $U_3$ .

Finally, the overall control laws for quadrotor's attitude system are given by

$$\begin{aligned}
 U_2 &= \mathbf{H}_\phi(\mathbf{r}_\phi, \mathbf{c}_\phi, \mathbf{a}_\phi) \boldsymbol{\beta}_\phi \\
 &\quad + \left( \frac{d_{\max} I_{xx}}{l} + \varepsilon_N \right) \text{sgn}(\mathbf{E}_\phi^T \mathbf{P}_\phi \mathbf{B}_\phi), \\
 U_3 &= \mathbf{H}_\theta(\mathbf{r}_\theta, \mathbf{c}_\theta, \mathbf{a}_\theta) \boldsymbol{\beta}_\theta \\
 &\quad + \left( \frac{d_{\max} I_{yy}}{l} + \varepsilon_N \right) \text{sgn}(\mathbf{E}_\theta^T \mathbf{P}_\theta \mathbf{B}_\theta), \\
 U_4 &= \mathbf{H}_\psi(\mathbf{r}_\psi, \mathbf{c}_\psi, \mathbf{a}_\psi) \boldsymbol{\beta}_\psi \\
 &\quad + (d_{\max} I_{zz} + \varepsilon_N) \text{sgn}(\mathbf{E}_\psi^T \mathbf{P}_\psi \mathbf{B}_\psi),
 \end{aligned} \tag{41}$$

where the subscripts  $\phi$ ,  $\theta$ , and  $\psi$  represent the roll, pitch, and yaw subsystem, respectively. To avoid chattering problem, the  $\text{sgn}(\cdot)$  function can be replaced by the saturation function  $\text{sat}(\cdot)$ .

## 5. Simulations

In this section, attitude and position control simulations are both implemented on a nonlinear quadrotor model to evaluate the performance of the proposed controller. We also compare our method to some other controllers to demonstrate the effectiveness of ELM. The parameters of the quadrotor for simulation are measured from a real platform as listed in Table 1.

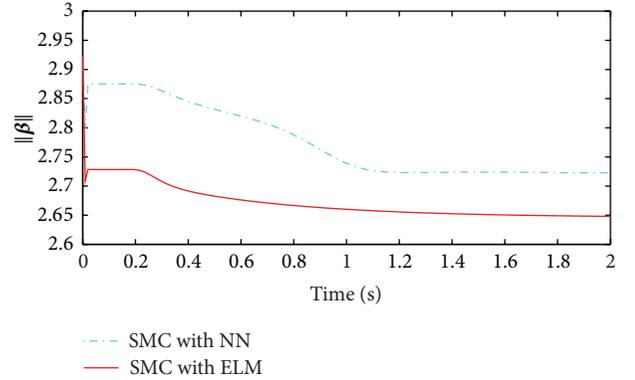
FIGURE 6: Learning curve of  $\|\boldsymbol{\beta}\|$  in attitude regulation control.

TABLE 1: Parameters of the quadrotor.

| Name                    | Parameter | Value      | Unit                       |
|-------------------------|-----------|------------|----------------------------|
| Mass                    | $m$       | 2.2        | kg                         |
| Inertia around $x$ axis | $I_{xx}$  | $1.676e-2$ | $\text{kg}\cdot\text{m}^2$ |
| Inertia around $y$ axis | $I_{yy}$  | $1.676e-2$ | $\text{kg}\cdot\text{m}^2$ |
| Inertia around $z$ axis | $I_{zz}$  | $2.314e-2$ | $\text{kg}\cdot\text{m}^2$ |
| Rotor inertia           | $J_r$     | 0.1        | $\text{kg}\cdot\text{m}^2$ |
| Arm length              | $l$       | 0.18       | m                          |

**5.1. Simulation Setup.** Two cases are tested for attitude control simulations. The first case is to regulate the quadrotor helicopter from an initial attitude ( $\phi_i = \pi/8$ ,  $\theta_i = \pi/8$ ,  $\psi_i = 0$ ) to a target attitude ( $\phi_t = -\pi/16$ ,  $\theta_t = -\pi/16$ ,  $\psi_t = 0$ ).

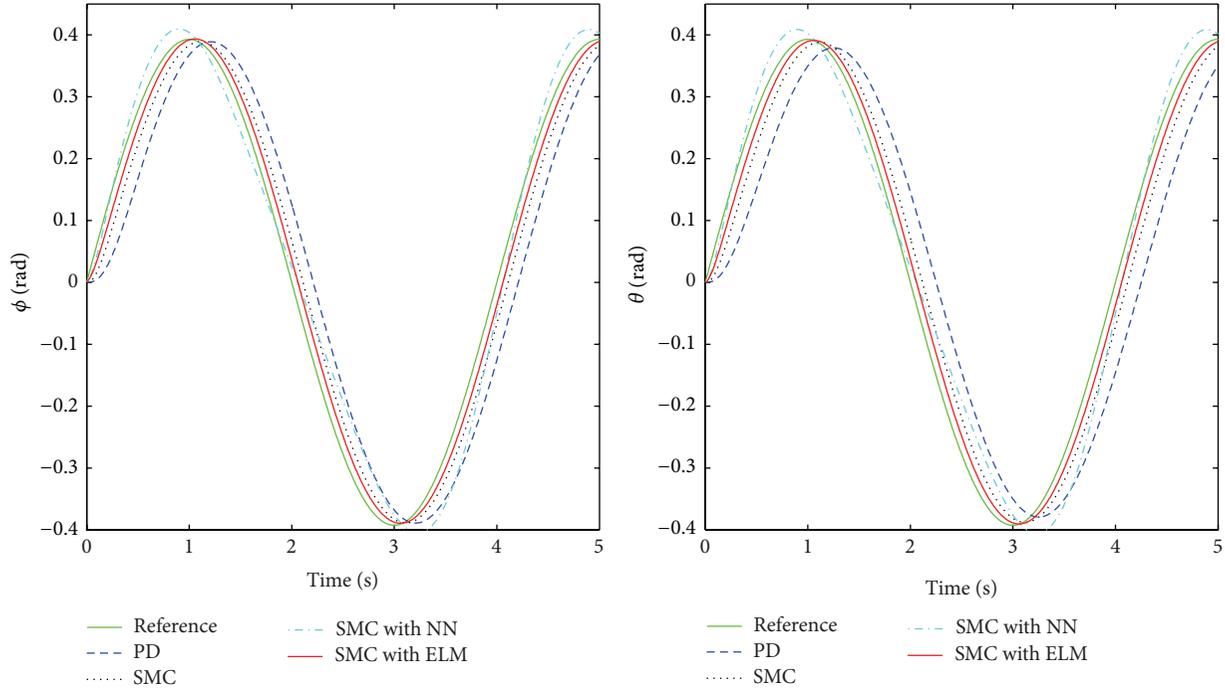


FIGURE 7: Roll and pitch angle tracking.

The attitude response speed and accuracy can be evaluated in this case. The second case is an attitude tracking mission. The quadrotor's attitude is set to follow a trajectory ( $\phi_d = (\pi/8) \sin(2\pi \times 0.25t)$ ,  $\theta_d = (\pi/8) \sin(2\pi \times 0.25t)$ ) while the yaw angle keeps fixed.

To further demonstrate the controller performance, position tracking simulations are also implemented, where the quadrotor is expected to follow a trajectory such as straight line and circle.

In all the cases above, the model uncertainty  $d$  in (22) is given by  $0.5 * \sin(\pi t)$ , and  $d_{\max} = 0.5$ . The controller parameters are chosen as follows:  $K_1 = 8$ ,  $K_2 = 128$ ,  $\mathbf{Q} = \text{diag}(100, 100)$ ,  $\eta = 0.05$ , and  $\varepsilon_N = 14$ . The number of hidden nodes  $L = 6$ . RBF nodes are selected as the hidden nodes whose parameter  $\mathbf{c}$  is generated in the interval  $[-5\pi/16, 5\pi/16]$  and  $a = 0.4$ .

To show the feasibility and advantage of the proposed controller, we compare our method (SMC with ELM) to a proportion-differentiation controller (PD), a standard sliding mode controller (SMC), and a SMC with back propagation based neural network (SMC with NN). The parameters of the PD controller are chosen as follows:  $k_{p\phi} = 5$ ,  $k_{d\phi} = 1.2$ ,  $k_{p\theta} = 5$ ,  $k_{d\theta} = 1.2$ ,  $k_{p\psi} = 5$ , and  $k_{d\psi} = 1$ . Sliding mode control gain  $G = 14.5$ . Sliding mode surface parameters  $c_1 = 8$  and  $c_2 = 1$ . The initial parameters of the SMC with NN are the same as the SMC with ELM.

**5.2. Results.** The results of roll ( $\phi$ ) and pitch ( $\theta$ ) angles regulation using different controllers are illustrated in Figure 4. As we can see, the proposed controller makes the attitude angles converge faster to the reference values than the other

controllers. PD controller has attitude overshoot problem which may result in vehicle's vibration. This problem can be avoided by decreasing PD control gains, but the response speed is sacrificed. The result from SMC with NN has large errors because of the slow convergence problem. The control inputs related to roll and pitch are shown in Figure 5. All of them are bounded and applicable. Inputs in our method are large at the beginning but decrease quickly. This is the reason why the response speed and accuracy are obtained simultaneously. In Figure 6, it indicates that the output weights  $\beta$  trained by ELM can converge faster than BP based neural networks. Thus, the contribution of ELM in quadrotor controller design is further confirmed.

Attitude tracking results are shown in Figure 7. ELM-assisted controller makes the attitude follow the reference trajectory pretty well while PD and SMC have a delay in tracking process. The result from SMC with NN also cannot follow the reference input well. It shows that the proposed controller has the best tracking performance among the four controllers. Tracking errors of this case are shown in Figure 8. We can see that our proposed method has the smallest error. Figure 9 shows the better convergency of output weights  $\beta$  using ELM than traditional neural networks.

The results of straight line tracking are shown in Figure 10 which demonstrate better tracking performance of proposed controller. As we can see from Figure 11, SMC with ELM has the smallest error compared to the others. In addition, the PD and SMC with NN cannot damp the errors quickly and have some oscillations. The learning curve of ELM also converges fast as shown in Figure 12.

Position tracking on a circle trajectory is shown in Figure 13. It seems that all methods can give a stable tracking

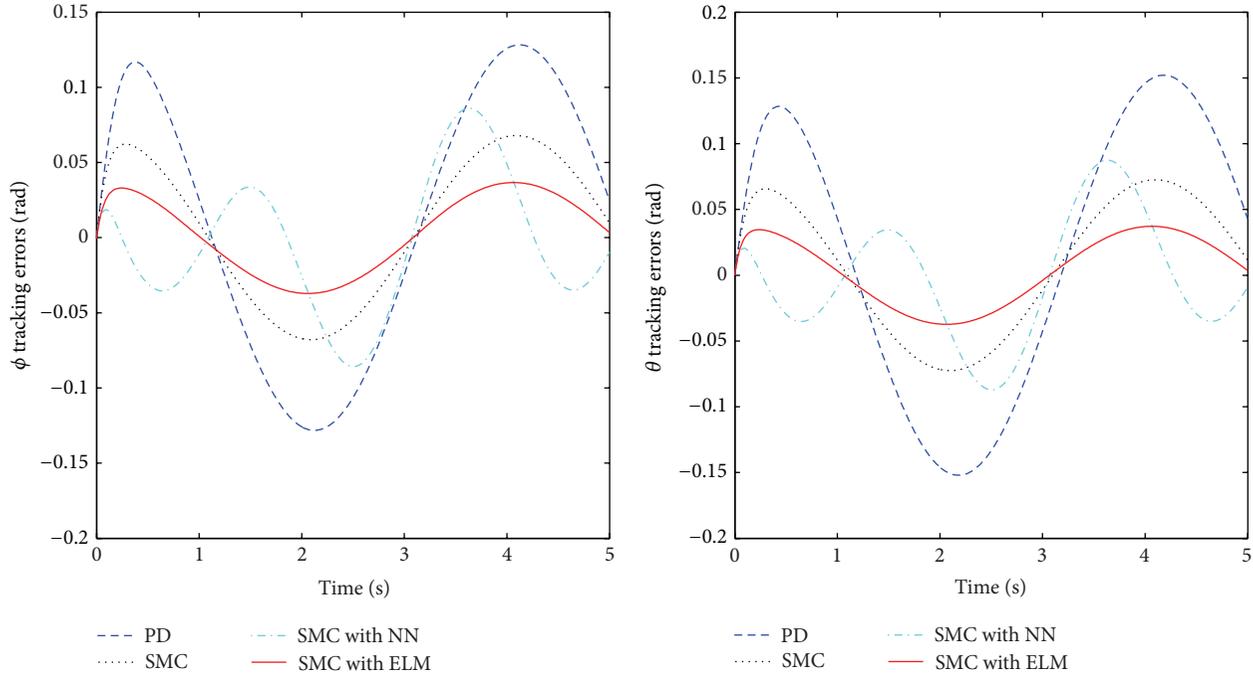


FIGURE 8: Attitude tracking errors.

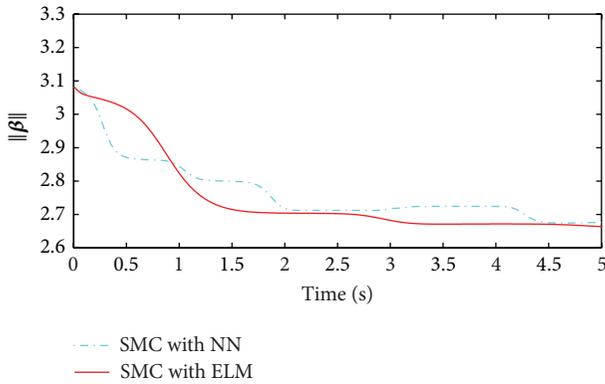


FIGURE 9: Learning curve of  $\|\beta\|$  in attitude tracking control.

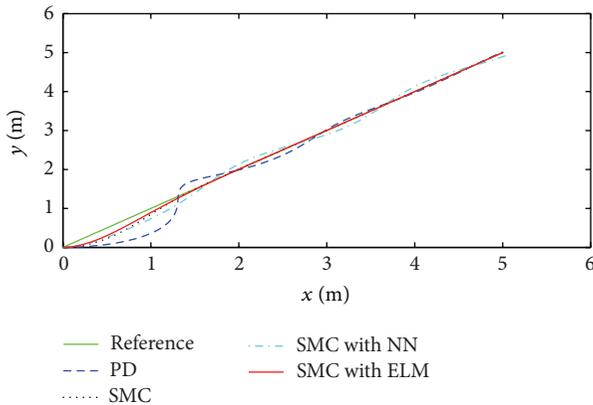


FIGURE 10: Position tracking on a straight line.

performance. However, as we can see, the SMC with ELM has the best performance in terms of stability and tracking accuracy. The training process of  $\beta$  in ELM is also better than BP based neural networks as given in Figure 14. ELM can converge quickly and remains unchanged afterwards in this test.

### 6. Conclusions

In this paper, an ELM-assisted adaptive controller combined with a sliding mode controller is designed for control of a quadrotor helicopter. A single hidden layer feedforward network whose input weights and hidden node parameters are generated randomly and fixed afterwards is used to approximate the unknown dynamic model and internal uncertainties. Different from the standard ELM algorithm, the output weights of this neural network are updated based on the Lyapunov second method to guarantee the stability of the attitude control system. A sliding mode controller is employed to compensate the approximation error of the SLFN and eliminate the external disturbances. Plenty of simulations on quadrotor's attitude and position control are implemented to validate the effectiveness of the proposed control scheme. The simulation results show that the proposed controller has better performance on response speed and control accuracy than simple PD controller. Furthermore, the comparison with the standard SMC and SMC with NN indicates that the extreme learning machine has potential ability to handle the unmodeled uncertainty problem in the control domain because of its fast convergence capability and good approximation ability.

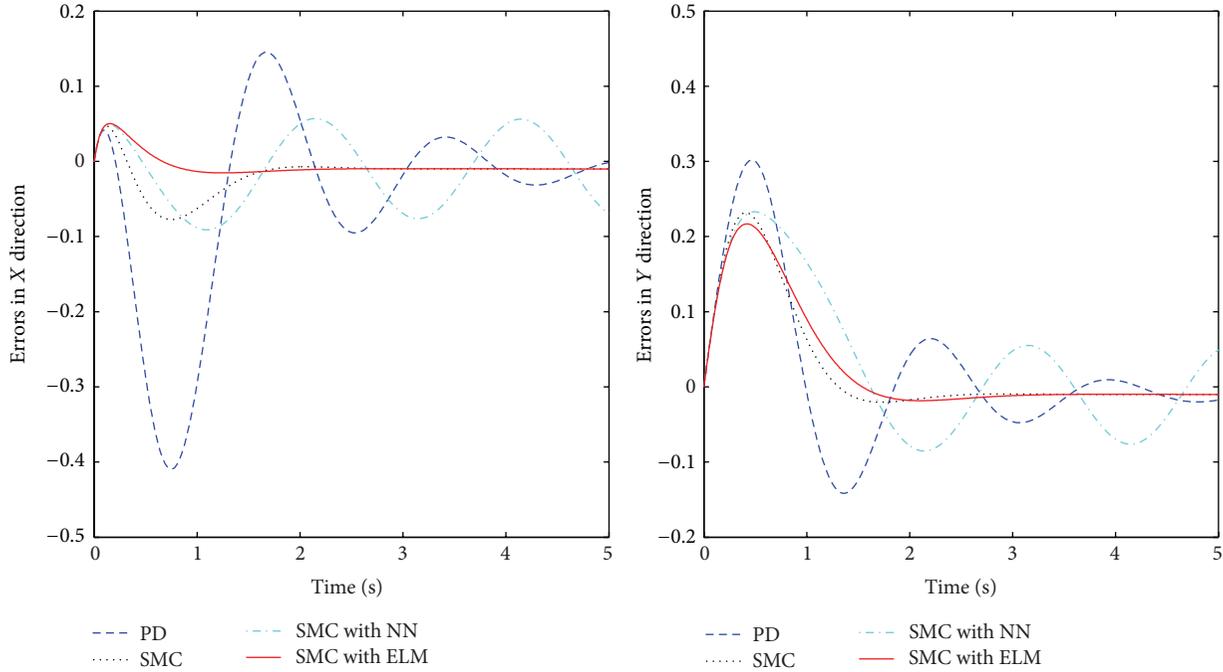


FIGURE 11: Tracking errors in X and Y direction.

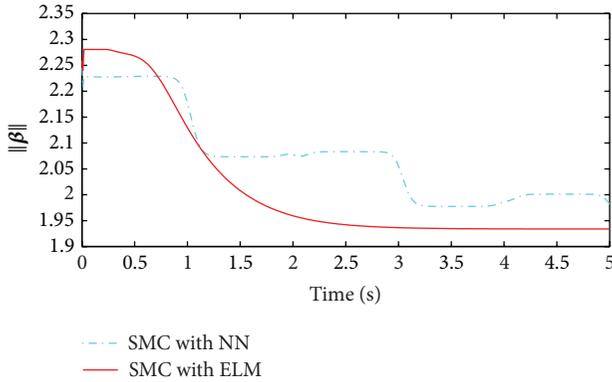


FIGURE 12: Learning curve of  $\|\beta\|$  in position tracking control (straight line).

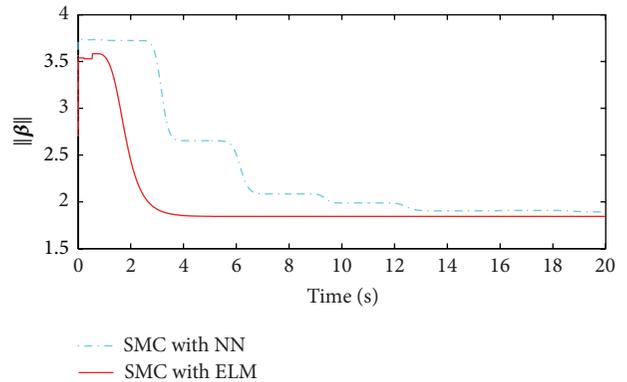


FIGURE 14: Learning curve of  $\|\beta\|$  in position tracking control (circle).

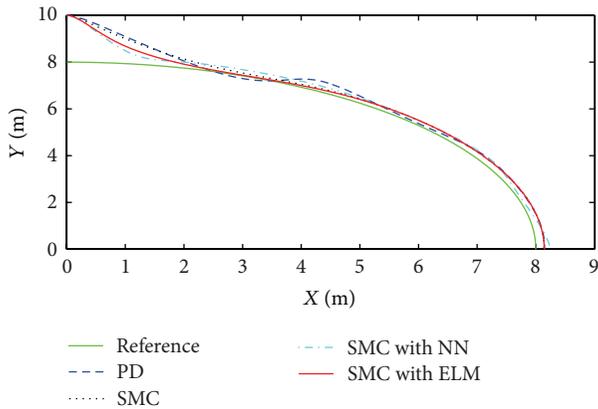


FIGURE 13: Position tracking on a circle trajectory.

For future work, the idea of composite design and reinforcement learning [28] could be borrowed to develop new ELM algorithm to achieve better results.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**Acknowledgments**

This work was supported by the National Natural Science Foundation of China (Grant no. 61005085) and Fundamental Research Funds for the Central Universities (2012QNA4024, N120408002).

## References

- [1] A. Nagaty, S. Saeedi, C. Thibault, M. Seto, and H. Li, "Control and navigation framework for quadrotor helicopters," *Journal of Intelligent and Robotic Systems*, vol. 70, no. 1–4, pp. 1–12, 2013.
- [2] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro Quadrotor," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pp. 2451–2456, IEEE, October 2004.
- [3] Y. Li and S. Song, "A survey of control algorithms for quadrotor unmanned helicopter," in *Proceedings of the IEEE 5th International Conference on Advanced Computational Intelligence (ICACI '12)*, pp. 365–369, October 2012.
- [4] S. Gonzalez-Vazquez and J. Moreno-Valenzuela, "A new nonlinear PI/PID controller for quadrotor posture regulation," in *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA '10)*, pp. 642–647, Morelos, Mexico, September–October 2010.
- [5] A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. S. Gaeid, "Flight PID controller design for a UAV quadrotor," *Scientific Research and Essays*, vol. 5, no. 23, pp. 3660–3667, 2010.
- [6] L. Minh and C. Ha, "Modeling and control of quadrotor MAV using vision-based measurement," in *Proceedings of the International Forum on Strategic Technology (IFOST '10)*, pp. 70–75, IEEE, October 2010.
- [7] N. G. Shakev, A. V. Topalov, O. Kaynak, and K. B. Shiev, "Comparative results on stabilization of the quad-rotor rotorcraft using bounded feedback controllers," *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1–4, pp. 389–408, 2012.
- [8] H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," in *Proceedings of the IEEE International Conference on Mechatronics (ICM '09)*, pp. 1–6, IEEE, April 2009.
- [9] B. Xu, F. Sun, H. Liu, and J. Ren, "Adaptive kriging controller design for hypersonic flight vehicle via back-stepping," *IET Control Theory & Applications*, vol. 6, no. 4, pp. 487–497, 2012.
- [10] B. Xu, X. Huang, D. Wang, and F. Sun, "Dynamic surface control of constrained hypersonic flight models with parameter estimation and actuator compensation," *Asian Journal of Control*, vol. 16, no. 1, pp. 162–174, 2014.
- [11] B. Xu, F. Sun, C. Yang, D. Gao, and J. Ren, "Adaptive discrete-time controller design with neural network for hypersonic flight vehicle via back-stepping," *International Journal of Control*, vol. 84, no. 9, pp. 1543–1552, 2011.
- [12] A. Das, F. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1–2, pp. 127–151, 2009.
- [13] A. Honglei, L. Jie, W. Jian, W. Jianwen, and M. Hongxu, "Backstepping-based inverse optimal attitude control of quadrotor," *International Journal of Advanced Robotic Systems*, vol. 10, article no. 223, 2013.
- [14] A. R. Patel, M. A. Patel, and D. R. Vyas, "Modeling and analysis of quadrotor using sliding mode control," in *Proceedings of the 44th Southeastern Symposium on System Theory (SSST '12)*, pp. 111–114, IEEE, March 2012.
- [15] J. Gadewadikar, F. L. Lewis, K. Subbarao, K. Peng, and B. M. Chen, "H-infinity static output-feedback control for rotorcraft," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 54, no. 4, pp. 629–646, 2009.
- [16] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear  $H_\infty$  control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.
- [17] C. Coza, C. Nicol, C. J. B. Macnab, and A. Ramirez-Serrano, "Adaptive fuzzy control for a quadrotor helicopter robust to wind buffeting," *Journal of Intelligent and Fuzzy Systems*, vol. 22, no. 5–6, pp. 267–283, 2011.
- [18] D. Gautam and C. Ha, "Control of a quadrotor using a smart self-tuning fuzzy PID controller," *International Journal of Advanced Robotic Systems*, vol. 10, article 380, 2013.
- [19] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor UAV using neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 50–66, 2010.
- [20] B. Xu, Z. Shi, C. Yang, and S. Wang, "Neural control of hypersonic flight vehicle model via time-scale decomposition with throttle setting constraint," *Nonlinear Dynamics*, vol. 73, no. 3, pp. 1849–1861, 2013.
- [21] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: onboard implementation and experimental results," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 279–284, IEEE, May 2012.
- [22] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [23] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [24] H.-J. Rong and G.-S. Zhao, "Direct adaptive neural control of nonlinear systems with extreme learning machine," *Neural Computing and Applications*, vol. 22, no. 3–4, pp. 577–586, 2013.
- [25] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Real-time learning capability of neural networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 863–878, 2006.
- [26] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [27] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, pp. 153–158, IEEE, October 2007.
- [28] B. Xu, C. Yang, and Z. Shi, "Reinforcement learning output feedback NN control using deterministic learning technique," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 635–641, 2014.

## Research Article

# A Robust AdaBoost.RT Based Ensemble Extreme Learning Machine

**Pengbo Zhang and Zhixin Yang**

*Department of Electromechanical Engineering, Faculty of Science and Technology, University of Macau, Macau*

Correspondence should be addressed to Zhixin Yang; [zyyang@umac.mo](mailto:zyyang@umac.mo)

Received 21 August 2014; Revised 12 November 2014; Accepted 13 November 2014

Academic Editor: Yi Jin

Copyright © 2015 P. Zhang and Z. Yang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Extreme learning machine (ELM) has been well recognized as an effective learning algorithm with extremely fast learning speed and high generalization performance. However, to deal with the regression applications involving big data, the stability and accuracy of ELM shall be further enhanced. In this paper, a new hybrid machine learning method called robust AdaBoost.RT based ensemble ELM (RAE-ELM) for regression problems is proposed, which combined ELM with the novel robust AdaBoost.RT algorithm to achieve better approximation accuracy than using only single ELM network. The robust threshold for each weak learner will be adaptive according to the weak learner's performance on the corresponding problem dataset. Therefore, RAE-ELM could output the final hypotheses in optimally weighted ensemble of weak learners. On the other hand, ELM is a quick learner with high regression performance, which makes it a good candidate of "weak" learners. We prove that the empirical error of the RAE-ELM is within a significantly superior bound. The experimental verification has shown that the proposed RAE-ELM outperforms other state-of-the-art algorithms on many real-world regression problems.

## 1. Introduction

In the past decades, computational intelligence methodologies are widely adopted and have been effectively utilized in various areas of scientific research and engineering applications [1, 2]. Recently, Huang et al. introduced an efficient learning algorithm, named extreme learning machine (ELM), for single-hidden layer feedforward neural networks (SLFNs) [3, 4]. Unlike conventional learning algorithms such as back-propagation (BP) methods [5] and support vector machines (SVMs) [6], ELM could randomly generate the hidden neuron parameters (the input weights and the hidden layer biases) before seeing the training data, and could analytically determine the output weights without tuning the hidden layer of SLFNs. As the random generated hidden neuron parameters are independent of the training data, ELM can reach not only the smallest training error but also the smallest norm of output weights. ELM overcomes several limitations in the conventional learning algorithms, such as local minimal and slow learning speed, and embodies very good generalization performance.

As a popular and pleasing learning algorithm, massive variants of ELM have been investigated in order to further improve its generalization performance. Rong et al. [7] proposed an online sequential fuzzy extreme learning machine (OS-Fuzzy-ELM) for function approximation and classification problems. Cao et al. [8] combined the voting based extreme learning machine [9] with online sequential extreme learning machine [10] into a new methodology, called voting based online sequential extreme learning machine (VOS-ELM). In addition, to solve the two drawbacks in the basic ELM, namely, the over-fitting problem and the unstable accuracy, Luo et al. [11] presented a novel algorithm, called sparse Bayesian extreme learning machine (SB-ELM), which estimates the marginal likelihood of the output weights automatically pruning the redundant hidden nodes. What is more, to overcome the limitations of supervised learning algorithms, according to the theory of semisupervised learning [12].

Although ELM has good generalization performances for classification and regression problems, how to efficiently perform training and testing on big data is challenging for

ELM as well. As a single learning machine, although ELM is quite stable compared to other learning algorithms, its classification and regression performance may still be slightly varied among different trails on big dataset. Many researchers sought for various ensemble methods that integrate a set of ELMs into a combined network structures, and verified that they could perform better than using individual ELM. Lan et al. [13] proposed an ensemble of online sequential ELM (EOS-ELM), which is comprised of several OS-ELM networks. The mean of the OS-ELM networks' outputs was used as the performance indicator of the ensemble networks. Liu and Wang [14] presented an ensemble-based ELM (EN-ELM) algorithm, where the cross-validation scheme was used to create an ensemble of ELM classifiers for decision making. Besides, Xue et al. [15] proposed a genetic ensemble of extreme learning machine (GE-ELM), which adopted genetic algorithms (GAs) to produce a group of candidate networks first. According to a specific ranking strategy, some of the networks were selected to ensemble a new network. More recently, Wang et al. [16] presented a parallelized ELM ensemble method based on  $M^3$ -network, called  $M^3$ -ELM. It could improve the computation efficiency by parallelism and solve imbalanced classification tasks through task decomposition.

To learn the exponentially increased number and types of data with high accuracy, the traditional learning algorithms may tend to suffer from overfitting problem. Hence, a robust and stable ensemble algorithm is of great importance. Dasarathy and Sheela [17] firstly introduced an ensemble system, whose idea is to partition the feature space using multiple classifiers. Furthermore, Hansen and Salamon [18] presented an ensemble of neural networks with a plurality consensus scheme to obtain far better performance in classification issues than approaching using single neural networks. After that, the ensemble-based algorithms have been widely explored [19–23]. Among the ensemble-based algorithms, Bagging and Boosting are the most prevailing methods for training neural network ensembles. The Bagging (short for Bootstrap Aggregation) algorithm randomly selects  $n$  bootstrap samples from  $N$  cardinality original training set ( $n < N$ ), and then the diversity in the bagging-based ensembles is ensured by the variations within the bootstrapped replicas on which each classifier is trained. By using relatively weak classifiers, the decision boundaries measurably vary with respect to relatively small perturbations in the training data. As an iterative method presented by Schapire [20] for generating a strong classifier, boosting could achieve arbitrarily low training error from an ensemble of weak classifiers, each of which can barely do better than random guessing. Whereafter, a novel boosting algorithm, called the adaptive boosting (AdaBoost), was presented by Schapire and Freund [21]. The AdaBoost algorithm makes improvement to traditional boosting methods in two perspectives. One is that the instances thereof are drawn into subsequent subdatasets from an iteratively updated sample distribution of the same training dataset. AdaBoost replaces randomly subsamples by weighted versions of the same training dataset which could be repeatedly utilized. The training dataset is therefore not required to be very large. Another is to define an ensemble

classifier through combination of weighted majority voting of a set of weak classifiers, where voting weights are based on classifiers' training errors.

However, many of the existing investigations on ensemble algorithms focus on classification problems. The ensemble algorithms on classification problems, unfortunately, cannot be directly applied on regression problems. Regression methods could provide predicated results through analyzing historical data. Forecasting and predication are important functional requirements for real-world applications, such as temperature prediction, inventory management, and positioning tracking in manufacturing execution system. To solve regression problems, based on the AdaBoost algorithm on the classification problem [24–26], Schapire and Freund [21] extended AdaBoost.M2 to AdaBoost.R. In addition, Drucker [27] proposed AdaBoost.R2 algorithm, which is based on ad hoc modification of AdaBoost.R. Besides, Avnimelech and Intrator [28] presented the notion of weak and strong learning and an appropriate equivalence theorem between them so as to improve the boosting algorithm in regression issues. What is more, Solomatine and Shrestha [29, 30] proposed a novel boosting algorithm, called as AdaBoost.RT. AdaBoost.RT projects the regression problems into the binary classification domain which could be processed by AdaBoost algorithm while filtering out those examples with the relative estimation error larger than the preset threshold value.

The proposed hybrid algorithm, which combines the effective learner, ELM, with the promising ensemble method, AdaBoost.RT algorithm, could inherit their intrinsic properties and shall be able to achieve good generalization performances for dealing with big data. Same as the development effort on general ensemble algorithms, the available ELM ensembles algorithms are mainly aimed at the classification problems, while the regression problems with ensemble algorithm have received relatively little attention. Tian and Mao [31] presented an ensemble ELM based on modified AdaBoost.RT algorithm (modified Ada-ELM) in order to predict the temperature of molten steel in ladle furnace. The novel hybrid learning algorithm combined the modified AdaBoost.RT with ELM, which possesses the advantage of ELM and overcomes the limitation of basic AdaBoost.RT by self-adaptively modifiable threshold value. The threshold value  $\Phi$  need not be constant; instead, it could be adjusted using a self-adaptive modification mechanism subjected to the change trend of the predication error at each iteration. The variation range of threshold value is set to be  $[0, 0.4]$ , as suggested by Solomatine and Shrestha [29, 30]. However, the initial value of  $\Phi$  is manually fixed to be the mean of the variation range of threshold value, ex.  $\Phi_0 = 0.2$ , according to an empirical suggestion. When one error rate  $\varepsilon_t$  is smaller than that in previous iteration,  $\varepsilon_{t-1}$ , the value of  $\Phi$  will decrease and vice versa. Hence, such empirical suggestion based method is not fully self-adaptive in the whole threshold domain. Moreover, the manually fixed initial threshold is not related to the properties of input dataset and the weak learners, which make the ensemble ELM hardly reach a generally optimized learning effect.

This paper presents a robust AdaBoost.RT based ensemble ELM (RAE-ELM) for regression problems, which combined ELM with the robust AdaBoost.RT algorithm. The robust AdaBoost.RT algorithm not only overcomes the limitation of the original AdaBoost.RT algorithm (original Ada-ELM), but also makes the threshold value of  $\Phi$  adaptive to the input dataset and ELM networks instead of presetting. The main idea of RAE-ELM is as follows. The ELM algorithm is selected as the “weak” learning machines to build the hybrid ensemble model. A new robust AdaBoost.RT algorithm is proposed to utilize the error statistics method to dynamically determine the regression threshold value rather than via manual selection which may only be ideal for very few regression cases. The mean and the standard deviation of the approximation errors will be computed at each iteration. The *robust threshold* for each weak learner is defined to be a scaled standard deviation. Based on the concept of standard deviation, those individual training data with error exceeding the robust threshold are regarded as “flaws in this training process” and shall be rejected. The rejected data will be processed in the late part of weak learners’ iterations.

We then analyze the convergence of the proposed robust AdaBoost.RT algorithm. It could be proved that the error of the final hypothesis output by the proposed ensemble algorithm,  $E_{\text{ensemble}}$ , is within a significantly superior bound. The proposed robust AdaBoost.RT based ensemble extreme learning machine can avoid overfitting because of the characteristic of ELM. ELM can tend to reach the solutions straightforwardly, and the error rate of regression outcome at each training process is much smaller than 0.5. Therefore, the proposed robust AdaBoost.RT based ensemble extreme learning machine selecting ELM as the “weak” learner can avoid overfitting. Moreover, as ELM is a fast learner with quite high regression performance, it contributes to the overall generalization performance of the robust AdaBoost.RT based ensemble module. The experiment results have demonstrated that the proposed robust AdaBoost.RT ensemble ELM (RAE-ELM) has superior learning properties in terms of stability and accuracy for regression issues and have better generalization performance than other algorithms.

This paper is organized as follows. Section 2 gives a brief review of basic ELM. Section 3 introduces the original and the proposed robust AdaBoost.RT algorithm. The hybrid robust AdaBoost.RT ensemble ELM (RAE-ELM) algorithm is then presented in Section 4. The performance evaluation of RAE-ELM and its regression ability are verified using experiments in Section 5. Finally, the conclusion is drawn in the last section.

## 2. Brief on ELM

Recently, Huang et al. [3, 4] proposed novel neural networks, called extreme learning machines (ELMs), for single-hidden layer feedforward neural networks (SLFNs) [32, 33]. ELM is based on the least-square method which could randomly assign the input weights and the hidden layer biases, and

then the output weights between the hidden nodes and the output layer can be analytically determined. Since the learning process in ELM can take place without iterative tuning, the ELM algorithm could tend to reach the solutions straightforwardly without suffering from those problems including local minimal, slow learning speed, and overfitting.

From the standard optimization theory point of view, the objective of ELM in minimizing both the training errors and the outputs weights can be presented as [4]

$$\text{Minimize: } L_{P_{\text{ELM}}} = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\boldsymbol{\xi}_i\|^2 \quad (1)$$

$$\text{Subject to: } \mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta} = \mathbf{t}_i^T - \boldsymbol{\xi}_i^T, \quad i = 1, \dots, N,$$

where  $\boldsymbol{\xi}_i = [\xi_{i,1}, \dots, \xi_{i,m}]^T$  is the training error vector of the  $m$  output nodes with respect to the training sample  $\mathbf{x}_i$ . According to KKT theorem, training ELM is equivalent to solving the dual optimization problem:

$$L_{D_{\text{ELM}}} = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\boldsymbol{\xi}_i\|^2 \quad (2)$$

$$- \sum_{i=1}^N \sum_{j=1}^m \alpha_{i,j} (\mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta}_j - t_{i,j} + \xi_{i,j}),$$

where  $\boldsymbol{\beta}_j$  is the vector of the weights between the hidden layer and the  $j$ th output node and  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_m]$ .  $C$  is the regularization parameter representing the trade-off between the minimization of training errors and the maximization of the marginal distance.

According to KKT theorem, we can obtain different solutions as follows.

(1) *Kernel Case.* Consider

$$\boldsymbol{\beta} = \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (3)$$

The ELM output function is

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \boldsymbol{\beta} = \mathbf{h}(\mathbf{x}) \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (4)$$

A kernel matrix for ELM is defined as follows:

$$\boldsymbol{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^T : \Omega_{\text{ELM},i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \quad (5)$$

Then, the ELM output function can be as follows:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \boldsymbol{\beta} = \mathbf{h}(\mathbf{x}) \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left( \frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\text{ELM}}^T \right)^{-1} \mathbf{T}. \quad (6)$$

In the special case, a corresponding kernel  $K(\mathbf{x}_i, \mathbf{x}_j)$  is used in ELM, instead of using the feature mapping  $\mathbf{h}(\mathbf{x})$

which need be known. We call  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}(\mathbf{x}_i)\mathbf{h}(\mathbf{x}_j)$  ELM random kernel, where the feature mapping  $\mathbf{h}(\mathbf{x})$  is randomly generated.

(2) *Nonkernel Case.* Similarly, based on KKT theorem, we have

$$\beta = \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{H}^T \mathbf{T}. \quad (7)$$

In this case, the ELM output function is

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \beta = \mathbf{h}(\mathbf{x}) \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{H}^T \mathbf{T}. \quad (8)$$

### 3. The Proposed Robust AdaBoost.RT Algorithm

We first describe the original AdaBoost.RT algorithm for regression problem and then present a new robust AdaBoost.RT algorithm in this section. The corresponding analysis on the novel algorithm will also be given.

3.1. *The Original AdaBoost.RT Algorithm.* Solomatine and Sherstha proposed AdaBoost.RT [29, 30], a new boost algorithm for regression problems, where the letters  $R$  and  $T$  represent regression and threshold, respectively. The original AdaBoost.RT algorithm is described as follows.

Input the following:

- (i) sequence of  $m$  examples  $(x_1, y_1), \dots, (x_m, y_m)$ , where output  $y \in R$ ,
- (ii) weak learning algorithm (weak learner),
- (iii) integer  $T$  specifying number of iterations (machines),
- (iv) threshold  $\Phi$  ( $0 < \Phi < 1$ ) for demarcating correct and incorrect predictions.

Initialize the following:

- (i) machine number or iteration  $t = 1$ ,
- (ii) distribution  $D_t(i) = 1/m$  for all  $i$ ,
- (iii) error rate  $\varepsilon_t = 0$ .

Learning steps (iterate while  $t \leq T$ ) are as follows.

*Step 1.* Call weak learner, providing it with distribution  $D_t$ .

*Step 2.* Build the regression model:  $f_t(x) \rightarrow y$ .

*Step 3.* Calculate absolute relative error (ARE) for each training example as

$$\text{ARE}_t(i) = \left| \frac{f_t(x_i) - y_i}{y_i} \right|. \quad (9)$$

*Step 4.* Calculate the error rate of  $f_t(x)$ :  $\varepsilon_t = \sum_{i: \text{ARE}_t(i) > \Phi} D_t(i)$ .

*Step 5.* Set  $\beta_t = \varepsilon_t^n$ , where  $n$  is power coefficient (e.g., linear, square, or cubic).

*Step 6.* Update distribution  $D_t$  as follows:

if  $\text{ARE}_t(i) \leq \Phi$ , then  $D_{t+1}(i) = (D_t(i)/Z_t) \times \beta_t$ ;  
else,  $D_{t+1}(i) = D_t(i)/Z_t$ ,  
where  $Z_t$  is a normalization factor chosen such that  $D_{t+1}$  will be a distribution.

*Step 7.* Set  $t = t + 1$ .

Output the following: the final hypotheses:

$$f_{\text{fin}}(x) = \frac{\sum_t \{(\log(1/\beta_t)) \times f_t(x)\}}{\sum_t (\log(1/\beta_t))}. \quad (10)$$

The AdaBoost.RT algorithm projects the regression problems into the binary classification domain. Based on the boosting regression estimators [28] and BEM [34], the AdaBoost.RT algorithm introduces the absolute relative error (ARE) to demarcate samples as either correct or incorrect predictions. If the ARE of any particular sample is greater than the threshold  $\Phi$ , the predicted value for this sample is regarded as the incorrect predictor. Otherwise, it is remarked as correct predictor. Such indication method is similar to the ‘‘misclassification’’ and ‘‘correct-classification’’ labeling used in classification problems. The algorithm will assign relatively large weights to those weak learners in the front of learner list that reach high correct prediction rate. The samples with incorrect prediction will be handled as ad hoc cases by the followed weak learners. The outputs from each weak learner are combined as the final hypotheses using the corresponding computed weights.

AdaBoost.RT algorithm requires manual selection of threshold  $\Phi$ , which is a main factor sensitively affecting the performance of committee machines. If  $\Phi$  is too small, very few samples will be treated as correct predictions which will easily get boosted. It requires the followed learners to handle a large number of ad hoc samples and make the ensemble algorithm unstable. On the other hand, if  $\Phi$  is too large, say, greater than 0.4, most of samples will be treated as correct predictions where they fail to reject those false samples. In fact, it will cause low convergence efficiency and overfitting. The initial AdaBoost.RT and its variant suffer the limitation in setting threshold value, which is specified either as a manually specified constant value or a variable changing in vicinity of 0.2. Both of their strategies are irrelevant to the regression capability of the weak learner. In order to determine the  $\Phi$  value effectively, a novel improvement of AdaBoost.RT is proposed in the following section.

3.2. *The Proposed Robust AdaBoost.RT Algorithm.* To overcome the limitations suffered by the current works on AdaBoost.RT, we embed the statistics theory into the AdaBoost.RT algorithm. It overcomes the difficulty to optimally determining the initial threshold value and enables the intermediate threshold values to be dynamically self-adjustable according to the intrinsic property of the input

data samples. The proposed robust AdaBoost.RT algorithm is described as follows.

(1) Input the following:

sequence of  $m$  samples  $(x_1, y_1), \dots, (x_m, y_m)$ ,  
 where output  $y \in R$ ,  
 weak learning algorithm (weak learner),  
 maximum number of iterations (machines)  $T$ .

(2) Initialize the following:

iteration index  $t = 1$ ,  
 distribution  $D_t(i) = 1/m$  for all  $i$ ,  
 the weight vector:  $w_i^t = D_t(i)$  for all  $i$ ,  
 error rate  $\varepsilon_t = 0$ .

(3) Iterate while  $t \leq T$  the following.

(1) Call weak learner,  $WL_t$ , providing it with distribution:

$$p^{(t)} = \frac{w^{(t)}}{Z_t}, \quad (11)$$

where  $Z_t$  is a normalization factor chosen such that  $p^{(t)}$  will be a distribution.

(2) Build the regression model:  $f_t(x) \rightarrow y$ .

(3) Calculate each error:  $e_t(i) = f_t(x_i) - y_i$ .

(4) Calculate the error rate:  $\varepsilon_t = \sum_{i \in P} p_i^{(t)}$ , where  $P = \{i \mid |e_t(i) - \bar{e}_t| > \lambda \sigma_t\}$ ,  $i \in [1, m]$ ,  $\bar{e}_t$  stands for the expected value, and  $\lambda \sigma_t$  is defined as robust threshold ( $\sigma_t$  stands for the standard deviation, the relative factor  $\lambda$  is defined as  $\lambda \in (0, 1)$ ).

If  $\varepsilon_t > 1/2$ , then set  $T = t - 1$  and abort loop.

(5) Set  $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ .

(6) Calculate contribution of  $f_t(x)$  to the final result:  $\alpha_t = -\log(\beta_t)$ .

(7) Update the weight vectors:

$$\begin{aligned} \text{if } i \notin P, \text{ then } w_i^{(t+1)} &= w_i^{(t)} \beta_t; \\ \text{else, } w_i^{(t+1)} &= w_i^{(t)}. \end{aligned}$$

(8) Set  $t = t + 1$ .

(4) Normalize  $\alpha_1, \dots, \alpha_T$ , such that  $\sum_{t=1}^T \alpha_t = 1$ .

$$\text{Output the final hypotheses: } f_{\text{fin}}(x) = \sum_{t=1}^T \alpha_t f_t(x).$$

At each iteration of the proposed robust AdaBoost.RT algorithm, the standard deviation of the approximation error distribution is used as a criterion. In the probability and statistics theory, the standard deviation measures the amount of variation or dispersion from the average. If the data points tend to be very close to the mean value, the standard deviation is low. On the other hand, if the data points are spread out

over a large range of values, a high standard deviation will be resulted in.

Standard deviation may be served as a measure of uncertainty for a set of repeated predictions. When deciding whether predictions agree with their correspondingly true values, the standard deviation of those predictions made by the underlined approximation function is of crucial importance: if the averaged distance from the predictions to the true values is large, then the regression model being tested probably needs to be revised. Because the sample points that fall outside the range of values could reasonably be expected to occur, the prediction accuracy rate of the model is low.

In the proposed robust AdaBoost.RT algorithm, the approximation error of  $t$ th weak learner,  $WL_t$ , for an input dataset could be represented as one statistics distribution with parameters  $\mu_t \pm \lambda \sigma_t$ , where  $\mu_t$  stands for the expected value,  $\sigma_t$  stands for the standard deviation, and  $\lambda$  is an adjustable relative factor that ranges from 0 to 1. The threshold value for  $WL_t$  is defined by the scaled standard deviation,  $\lambda \sigma_t$ . In the hybrid learning algorithm, the trained weak learners are assumed to be able to generate small prediction error ( $\varepsilon_t < 1/2$ ). For all  $\mu_t, \mu_t \in (0 - \eta, 0 + \eta)$ ,  $\lim \eta \rightarrow 0$ , and  $t = 1, \dots, T$ , where  $\eta$  denotes a small error limit approaching zero. The population mean of a regression error distribution is closer to the targeted zero than elements in the population. Therefore, the means of the obtained regression errors are fluctuating around zero within a small range. The standard deviation,  $\sigma_t$ , is solely determined by the individual samples and the generalization performance of the weak learner  $WL_t$ . Generally,  $\sigma_t$  is relatively large such that most of the outputs will be located within the range  $[-\sigma_t, +\sigma_t]$ , which tends to make the boosting process unstable. To maintain a stable adjusting of the threshold value, a relative factor  $\lambda$  is applied on the standard deviation,  $\sigma_t$ , which results in the robust threshold,  $\lambda \sigma_t$ . For those samples that fall within the threshold range  $[-\lambda \sigma_t, +\lambda \sigma_t]$ , they are treated as "accepted" samples. Other samples are treated as "rejected" samples. With the introduction of the robust threshold, the algorithm will be stable and resistant to noise in the data. According to the error rate  $\varepsilon_t$  of each weak learner's regression model, each weak learner  $WL_t$  will be assigned with one accordingly computed weight  $\alpha_t$ .

For one regression problem, the performances of different weak learners may be different. The regression error distributions for different weak learners under robust threshold are shown in Figure 1.

In Figure 1(a), the weak learner  $WL_t$  generates a regression error distribution with large error rate, where the standard deviation is relatively large. On the other hand, another weak learner  $WL_{t+j}$  may generate an error distribution with small standard deviations as shown in Figure 1(b). Suppose red triangle points represent "rejected" samples, whose regression error rate is greater than the specified robust threshold,  $\varepsilon_t = \sum_{i \in P} p_i^{(t)}$ , where  $P = \{i \mid |e_t(i) - \bar{e}_t| > \lambda \sigma_t\}$ ,  $i \in [1, m]$ , as described in Step (4) of the proposed algorithm. The green circular points represent those "accepted" samples, which own regression errors less than the robust threshold. The weight vector  $w_i^t$  for every "accepted" sample will be

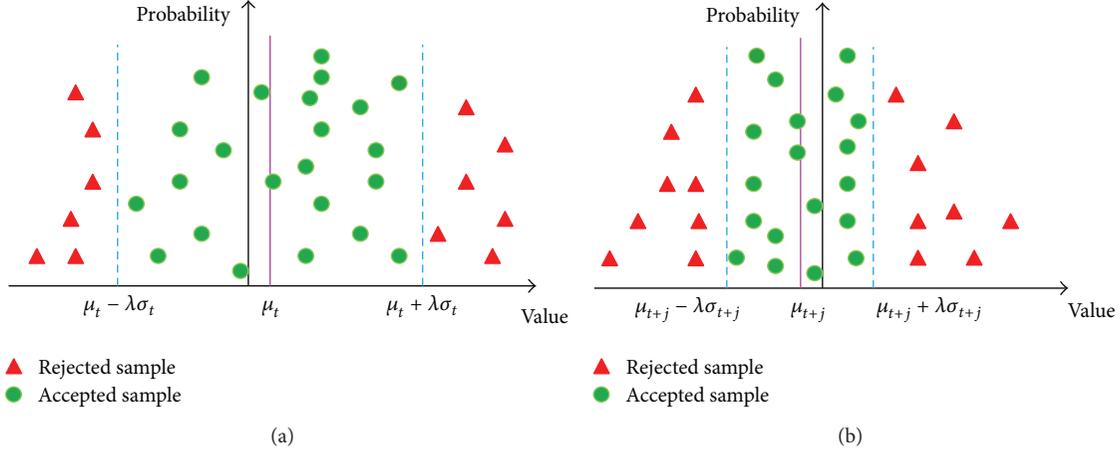


FIGURE 1: Different regression error distributions for different weak learners under robust threshold. (a) Error distribution of weak learner  $WL_t$  that results in a large threshold. (b) Error distribution of weak learner  $WL_{t+j}$  that results in a small threshold.

dynamically changed for each weak learner  $WL_t$ , while that for “rejected” samples will be unchanged.

As illustrated in Figure 1, in terms of stability and accuracy, the regression capability of weak learner  $WL_{t+j}$  is superior to  $WL_t$ . The robust threshold values for  $WL_{t+j}$  and  $WL_t$  are computed respectively, where the former is smaller than the latter, to discriminate their correspondingly different regression performances. The proposed method overcomes the limitation suffered by the existing methods where the threshold value is set empirically. The critical factor used in the boosting process, the threshold, becomes robust and self-adaptive to the individual weak learners’ performance on the input data samples. Therefore, the proposed robust AdaBoost.RT algorithm is capable to output the final hypotheses in optimally weighted ensemble of the weak learners.

In the following, we show that the training error of the proposed robust AdaBoost.RT algorithm is as bounded. One lemma needs to be given in order to prove the convergence of this algorithm.

**Lemma 1.** *The convexity argument,  $x^d \leq 1 - (1 - x)d$ , holds for any  $x \geq 0$  and  $0 \leq d \leq 1$ .*

**Theorem 2.** *The improved adaptive AdaBoost.RT algorithm generates hypotheses with errors  $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_T < 1/2$ . Then, the error  $E_{ensemble}$  of the final hypothesis output by this algorithm is bounded above by*

$$E_{ensemble} \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t (1 - \varepsilon_t)}. \quad (12)$$

*Proof.* In this proof, we need to transform the regression problem  $X \rightarrow Y$  into binary classification problems  $\{X, Y\} \rightarrow \{0, 1\}$ . In the proposed improved adaptive AdaBoost.RT algorithm, the mean of errors ( $\bar{\varepsilon}$ ) is assumed to be closed to zero. Thus, the dynamically adaptive thresholds can ignore the mean of errors.

Let  $I_t = \{i \mid |f_t(x_i) - y_i| < \lambda\sigma_t\}$ , if  $i \notin I_t$   $I_t^i = 1$ ; otherwise,  $I_t^i = 0$ .

The final hypothesis output  $f$  makes a mistake on sample  $i$  only if

$$\prod_{t=1}^T \beta_t^{-I_t^i} \geq \left( \prod_{t=1}^T \beta_t \right)^{-1/2}. \quad (13)$$

The final weight of any sample  $i$  is

$$w_i^{T+1} = D(i) \prod_{t=1}^T \beta_t^{(1-I_t^i)}. \quad (14)$$

Combining (13) and (14), the sum of the final weights is bounded by the sum of the final weights of rejected samples. Consider

$$\begin{aligned} \sum_{i=1}^m w_i^{T+1} &\geq \sum_{i \notin I_{T+1}^i} w_i^{T+1} \geq \left( \sum_{i \notin I_{T+1}^i} D(i) \right) \left( \prod_{t=1}^T \beta_t \right)^{1/2} \\ &= E_{ensemble} \left( \prod_{t=1}^T \beta_t \right)^{1/2}, \end{aligned} \quad (15)$$

where the  $E_{ensemble}$  is the error of the final hypothesis output. Based on Lemma 1,

$$\begin{aligned} \sum_{i=1}^m w_i^{t+1} &= \sum_{i=1}^m w_i^t \beta_t^{1-I_t^i} \leq \sum_{i=1}^m w_i^t (1 - (1 - \beta_t)(1 - I_t^i)) \\ &= \left( \sum_{i=1}^m w_i^t \right) (1 - (1 - \varepsilon_t)(1 - \beta_t)). \end{aligned} \quad (16)$$

Combining those inequalities for  $t = 1, \dots, T$ , the following equation could be obtained:

$$\sum_{i=1}^m w_i^{T+1} \leq \prod_{t=1}^T (1 - (1 - \varepsilon_t)(1 - \beta_t)). \quad (17)$$

Combining (15) and (17), we obtain that

$$E_{\text{ensemble}} \leq \prod_{t=1}^T \frac{1 - (1 - \varepsilon_t)(1 - \beta_t)}{\sqrt{\beta_t}}. \quad (18)$$

Considering that all factors in multiplication are positive, the minimization of the right hand side could be resorted to compute the minimization of each factor individually.  $\beta_t$  could be computed as  $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$  when setting the derivative of the  $t$ th factor to be zero. Substitute this computed  $\beta_t$  into (18), completing the proof.  $\square$

Unlike the original AdaBoost.RT and its existent variants, the robust threshold in the proposed AdaBoost.RT algorithm is determined and could be self-adaptively adjusted according to the individual weak learners and data samples. Through the analysis on the convergence of the proposed robust AdaBoost.RT algorithm, it could be proved that the error of the final hypothesis output by the proposed ensemble algorithm,  $E_{\text{ensemble}}$ , is within a significantly superior bound. The study shows that the robust AdaBoost.RT algorithm proposed in this paper can overcome the limitations existing in the available AdaBoost.RT algorithms.

#### 4. A Robust AdaBoost.RT Ensemble-Based Extreme Learning Machine

In this paper, a robust AdaBoost.RT ensemble-based extreme learning machine (RAE-ELM), which combines ELM with the robust AdaBoost.RT algorithm described in previous section, is proposed to improve the robustness and stability of ELM. A set of  $T$  number of ELMs is adopted as the “weak” learners. In the training phase, the RAE-ELM utilizes the proposed robust AdaBoost.RT algorithm to train every ELM model and assign an ensemble weight accordingly, in order that each ELM achieves corresponding distribution based on the training output. The optimally weighted ensemble model of ELMs,  $E_{\text{ensemble}}$ , is the final hypothesis output used for making prediction on testing dataset. The proposed RAE-ELM is illustrated as follows in Figure 2.

**4.1. Initialization.** For the first weak learner,  $\text{ELM}_1$  is supplied with  $m$  training samples with the uniformed distribution of weights in order that each sample owns equal opportunity to be chosen during the first training process for  $\text{ELM}_1$ .

**4.2. Distribution Updating.** The relative prediction error rates are used to evaluate the performance of this ELM. The prediction error of  $t$ th ELM,  $\text{ELM}_t$ , for the input data samples could be represented as one statistics distribution,  $\mu_t \pm \lambda\sigma_t$ , where  $\mu_t$  stands for the expected value, and  $\lambda\sigma_t$  is defined as robust threshold ( $\sigma_t$  stands for the standard deviation,

and the relative factor  $\lambda$  is defined as  $\lambda \in (0, 1)$ ). The robust threshold is applied to demarcate prediction errors as “accepted” or “rejected.” If the prediction error of one particular sample falls into the region  $\mu_t \pm \lambda\sigma_t$  that is bounded by the robust thresholds, the prediction of this sample is regarded as “accepted” for  $\text{ELM}_t$  and vice versa for “rejected” predictions. The probabilities of the “rejected” predictions are accumulated to calculate the error rate  $\varepsilon_t$ . ELM attempts to achieve the  $f_t(x)$  with small error rate.

The robust AdaBoost.RT algorithm will calculate the distribution for next  $\text{ELM}_{t+1}$ . For every sample that is correctly predicted by the current  $\text{ELM}_t$ , the corresponding weight vector will be multiplied by the error rate function  $\beta_t$ . Otherwise, the weight vector remains unchanged. Such process will be iterated for the next  $\text{ELM}_{t+1}$  till the last learner  $\text{ELM}_T$ , unless  $\varepsilon_t$  is higher than 0.5. Because once the error rate is higher than 0.5, the AdaBoost algorithm does not converge and tends to overfitting [21]. Hence, the error rate must be less than 0.5.

**4.3. Decision Making on RAE-ELM.** The weight updating parameter  $\beta_t$  is used as an indicator of regression effectiveness of the  $\text{ELM}_t$  in the current iteration. According to the relationship between  $\varepsilon_t$  and  $\beta_t$ , if  $\varepsilon_t$  increases,  $\beta_t$  will become larger as well. The RAE-ELM will grant a small ensemble weight for the  $\text{ELM}_t$ . On the other hand, the  $\text{ELM}_t$  with relatively superior regression performance will be granted with a larger ensemble weight. The hybrid RAE-ELM model combines the set of ELMs under different weights as the final hypothesis for decision making.

#### 5. Performance Evaluation of RAE-ELM

In this section, the performance of the proposed RAE-ELM learning algorithm is compared with other popular algorithms on 14 real-world regression problems covering different domains from UCI Machine Learning Repository [35], whose specifications of benchmark datasets are shown in Table 1. The ELM based algorithms to be compared include basic ELM [4], original AdaBoost.RT based ELM (original Ada-ELM) [30], the modified self-adaptive AdaBoost.RT ELM (modified Ada-ELM) [31], support vector regression (SVR) [36], and least-square support vector regression (LS-SVR) [37]. All the evaluations are conducted in Matlab environment running on a Windows 7 machine with 3.20 GHz CPU and 4 GB RAM.

In our experiments, all the input attributes are normalized into the range of  $[-1, 1]$ , while the outputs are normalized into  $[0, 1]$ . As the real-world benchmark datasets are embedded with noise and their distributions are unknown, which are of small sizes, low dimensions, large sizes, and high dimensions, for each trial of simulations, the whole data set of the application is randomly partitioned into training dataset and testing dataset with the number of samples shown in Table 1. 25% of the training data samples are used as the validation dataset. Each partitioned training, validation, and testing dataset will be kept fixed as inputs for all these algorithms.

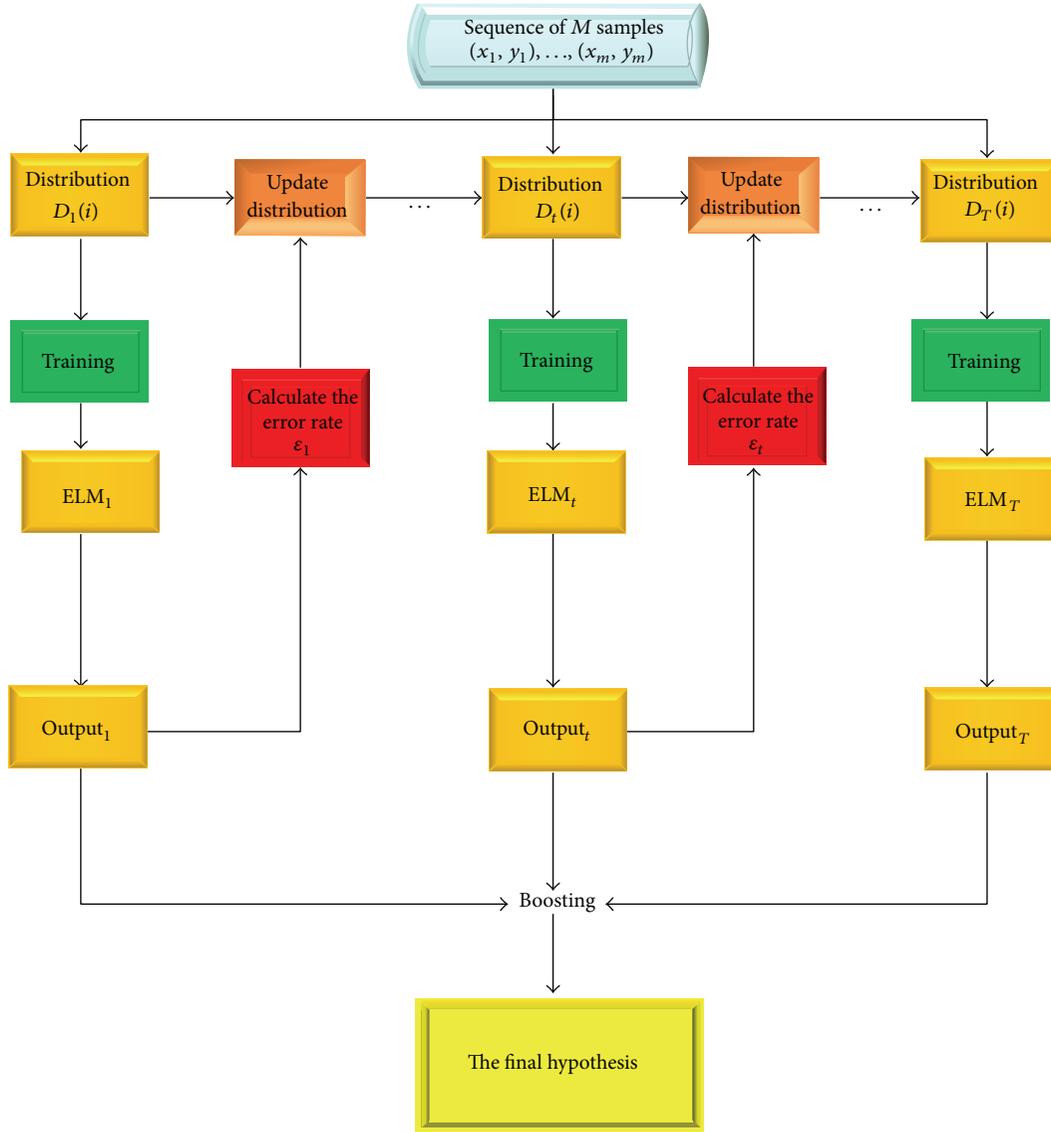


FIGURE 2: Block diagram of RAE-ELM.

For RAE-ELM, basic ELM, original Ada-ELM, and modified Ada-ELM algorithms, the suitable numbers of hidden nodes of them are determined using the preserved validation dataset, respectively. The sigmoid function  $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$  is selected as the activation function in all the algorithms. Fifty trails of simulations have been conducted for each problem, with training, validation, and testing samples randomly split for each trail. The performances of the algorithms are verified using the average root mean square error (RMSE) in testing. The significantly better results are highlighted in boldface.

**5.1. Model Selection.** In ensemble algorithms, the number of networks in the ensemble needs to be determined. According to Occam's Razor theory, excessively complex models are affected by statistical noise, whereas simpler models may capture the underlying structure better and may thus have

better predictive performance. Therefore, the parameter,  $T$ , which is the number of weak learners need not be very large.

We define  $\epsilon_t$  in (12) as  $\epsilon_t = 0.5 - \gamma_t$ , where  $\gamma_t > 0$  is constant, it results in

$$E_{\text{ensemble}} \leq \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2} = e^{(-\sum_{t=1}^T \text{KL}(1/2 \| 1/2 - \gamma_t))} \leq e^{(-2 \sum_{t=1}^T \gamma_t^2)}, \quad (19)$$

where KL is the *Kullback-Leibler divergence*.

We then simplify (19) by using  $0.5 - \gamma$  instead of  $0.5 - \gamma_t$ , that is, each  $\gamma_t$  is set to be the same. We can get

$$E_{\text{ensemble}} \leq (1 - 4\gamma^2)^{T/2} = e^{(-T * \text{KL}(1/2 \| 1/2 - \gamma))} \leq e^{(-2T\gamma^2)}. \quad (20)$$

TABLE 1: Specification of real-world regression benchmark datasets.

| Problems          | #Observations |              | #Attributes |
|-------------------|---------------|--------------|-------------|
|                   | Training data | Testing data |             |
| LVST              | 90            | 35           | 308         |
| Yeast             | 837           | 645          | 7           |
| Computer hardware | 110           | 99           | 7           |
| Abalone           | 3150          | 1027         | 7           |
| Servo             | 95            | 72           | 4           |
| Parkinson disease | 3000          | 2875         | 21          |
| Housing           | 350           | 156          | 13          |
| Cloud             | 80            | 28           | 9           |
| Auto price        | 110           | 49           | 9           |
| Breast cancer     | 100           | 94           | 32          |
| Balloon           | 1500          | 833          | 4           |
| Auto-MPG          | 300           | 92           | 7           |
| Bank              | 5000          | 3192         | 8           |
| Census (house8L)  | 15000         | 7784         | 8           |

From (20), we can obtain the upper bound number of iterations of this algorithm. Consider

$$T \leq \left\lceil \frac{1}{2\gamma^2} \ln \frac{1}{E_{\text{ensemble}}} \right\rceil. \quad (21)$$

For RAE-ELM, the number of ELM networks need be determined. The number of ELM networks is set to be 5, 10, 15, 20, 25, and 30 in our simulations, and the optimal parameter is selected as the one which results in the best average RMSE in testing.

Besides, in our simulation trails, the relative factor  $\lambda$  in RAE-ELM is the parameter which needs to be optimized within the range  $\lambda \in (0, 1)$ . We start simulations for  $\lambda$  at 0.1 and increase them at the interval of 0.1. Table 2 shows the examples of setting both  $T$  and  $\lambda$  for our simulation trail.

As illustrated in Table 2 and Figure 3, RAE-ELM with sigmoid activation function could achieve good generalization performance for Parkinson disease dataset as long as the number of ELM networks  $T$  is larger than 15. For a given number of ELM networks, RMSE is less sensitive to the variation of  $\lambda$  and tends to be smaller when  $\lambda$  is around 0.5. For a fair comparison, we set RAE-ELM with  $T = 20$  and  $\lambda = 0.5$  in the following experiments. For both original Ada-ELM and modified Ada-ELM, when the number of ELM networks is less than 7, the ensemble model is unstable. The number of ELM networks is also set to be 20 for both original Ada-ELM and modified Ada-ELM.

We use the popular Gaussian kernel function  $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$  in both SVR and LS-SVR. As is known to all, the performances of SVR and LS-SVR are sensitive to the combinations of  $(C, \gamma)$ . Hence, the cost parameter  $C$  and the kernel parameter  $\gamma$  need to be adjusted appropriately in a wide range so as to obtain good generalization performances. For each data set, 50 different values of  $C$  and 50 different values of  $\gamma$ , that is, 2500 pairs of  $(C, \gamma)$ , are applied as the adjustment parameters. The different values of  $C$  and  $\gamma$  are  $\{2^{-24}, 2^{-23}, \dots, 2^{24}, 2^{25}\}$ . In both SVR and LS-SVR, the best

TABLE 2: Performance of proposed RAE-ELM with different values of  $T$  and  $\lambda$  for Parkinson disease dataset.

| $\lambda$  | $T$    |        |        |        |        |        |
|------------|--------|--------|--------|--------|--------|--------|
|            | 5      | 10     | 15     | 20     | 25     | 30     |
| <b>0.1</b> | 0.2472 | 0.2374 | 0.2309 | 0.2259 | 0.2251 | 0.2249 |
| <b>0.2</b> | 0.2469 | 0.2315 | 0.2276 | 0.2241 | 0.2249 | 0.2245 |
| <b>0.3</b> | 0.2447 | 0.2298 | 0.2248 | 0.2235 | 0.2230 | 0.2233 |
| <b>0.4</b> | 0.2426 | 0.2287 | 0.2249 | 0.2227 | 0.2224 | 0.2227 |
| <b>0.5</b> | 0.2428 | 0.2296 | 0.2251 | 0.2219 | 0.2216 | 0.2215 |
| <b>0.6</b> | 0.2427 | 0.2298 | 0.2246 | 0.2221 | 0.2214 | 0.2213 |
| <b>0.7</b> | 0.2422 | 0.2285 | 0.2257 | 0.2229 | 0.2217 | 0.2219 |
| <b>0.8</b> | 0.2441 | 0.2301 | 0.2265 | 0.2228 | 0.2225 | 0.2228 |
| <b>0.9</b> | 0.2461 | 0.2359 | 0.2312 | 0.2243 | 0.2237 | 0.2230 |

performed combinations of  $(C, \gamma)$  are selected for each data set as presented in Table 3.

For basic ELM and other ELM-based ensemble methods, the sigmoid function  $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$  is selected as the activation function in all the algorithms. The parameters  $(C, L)$  need be selected so as to achieve the best generalization performance, where the cost parameter  $C$  is selected from the range  $\{2^{-24}, 2^{-23}, \dots, 2^{24}, 2^{25}\}$  and the different values of the hidden nodes  $L$  are  $\{10, 20, \dots, 1000\}$ .

In addition, for the original AdaBoost.RT-based ensemble ELM, the threshold  $\Phi$  should be chosen before seeing the datasets. In the original AdaBoost.RT-based ensemble ELM, the threshold  $\Phi$  is required to be manually selected according to an empirical suggestion, which is a sensitive factor affecting the regression performance. If  $\Phi$  is too low, then it is generally difficult to obtain a sufficient number of “accepted” samples. However if  $\Phi$  is too high, some wrong samples are treated as “accepted” ones and the ensemble model tends to be unstable. According to Shrestha and Solomatine’s experiments, the threshold  $\Phi$  shall be defined between 0 and 0.4 in order to make the ensemble model stable [30]. In our simulations, we incrementally set thresholds within the range from 0 to 0.4. The original Ada-ELM with threshold values at  $\{0.1, 0.15, \dots, 0.35, 0.4\}$  could generate satisfied results for all the regression problems, where the best performed original Ada-ELM is shown in boldface. What is more, the modified Ada-ELM algorithm needs to select an initial value of  $\Phi_0$  to calculate the followed thresholds in the iterations. Tian and Mao suggested setting the default initial value of  $\Phi_0$  to be 0.2 [31]. Considering that the manually fixed initial threshold is not related to the characteristics of ELM prediction effect on input dataset, the algorithm may not reach the best generalization performance. In our simulations, we compare the performances of different modified Ada-ELMs at correspondingly different initial threshold values  $\Phi_0$  set to be  $\{0.1, 0.15, \dots, 0.3, 0.35\}$ . The best performed modified Ada-ELM is also presented in Table 3 as well.

*5.2. Performance Comparisons between RAE-ELM and Other Learning Algorithms.* In this subsection, the performance of the proposed RAE-ELM is compared with other learning algorithms, including basic ELM [4], original Ada-ELM [30], and modified Ada-ELM [31], support vector regression [36],

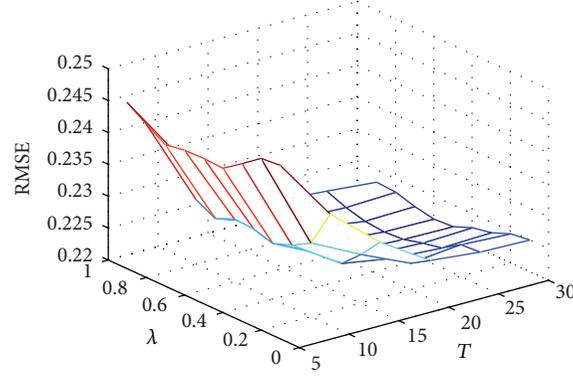


FIGURE 3: The average testing RMSE with different values of  $T$  and  $\lambda$  in RAE-ELM for Parkinson disease dataset.

TABLE 3: Parameters of RAE-ELM and other learning algorithms.

| Datasets          | RAE ELM<br>( $C, L$ ) | Basic ELM [4]<br>( $C, L$ ) | Original AdaELM [30]<br>( $C, L, \phi$ ) | Modified AdaELM [31]<br>( $C, L, \phi_0$ ) | SVR [36]<br>( $C, \gamma$ ) | LS SVR [37]<br>( $C, \gamma$ ) |
|-------------------|-----------------------|-----------------------------|------------------------------------------|--------------------------------------------|-----------------------------|--------------------------------|
| LVST              | $(2^{13}, 960)$       | $(2^5, 710)$                | $(2^{-7}, 530, 0.3)$                     | $(2^{-7}, 460, 0.35)$                      | $(2^{17}, 2^2)$             | $(2^8, 2^0)$                   |
| Yeast             | $(2^{10}, 540)$       | $(2^1, 340)$                | $(2^5, 710, 0.25)$                       | $(2^{-1}, 800, 0.2)$                       | $(2^{-7}, 2^0)$             | $(2^{-9}, 2^3)$                |
| Computer hardware | $(2^{-4}, 160)$       | $(2^{11}, 530)$             | $(2^{16}, 240, 0.35)$                    | $(2^{10}, 170, 0.25)$                      | $(2^{-9}, 2^{21})$          | $(2^0, 2^6)$                   |
| Abalone           | $(2^0, 150)$          | $(2^0, 200)$                | $(2^{-22}, 330, 0.2)$                    | $(2^6, 610, 0.15)$                         | $(2^3, 2^{18})$             | $(2^1, 2^{14})$                |
| Servo             | $(2^1, 340)$          | $(2^{-6}, 650)$             | $(2^0, 110, 0.3)$                        | $(2^9, 230, 0.2)$                          | $(2^{-2}, 2^2)$             | $(2^4, 2^{-3})$                |
| Parkinson disease | $(2^9, 830)$          | $(2^{-8}, 460)$             | $(2^4, 370, 0.35)$                       | $(2^{-4}, 590, 0.25)$                      | $(2^1, 2^0)$                | $(2^1, 2^7)$                   |
| Housing           | $(2^{-19}, 270)$      | $(2^{-9}, 310)$             | $(2^{-1}, 440, 0.25)$                    | $(2^{17}, 220, 0.2)$                       | $(2^{21}, 2^{-4})$          | $(2^3, 2^{12})$                |
| Cloud             | $(2^{-5}, 590)$       | $(2^{13}, 880)$             | $(2^0, 1000, 0.2)$                       | $(2^2, 760, 0.25)$                         | $(2^7, 2^5)$                | $(2^4, 2^{-11})$               |
| Auto price        | $(2^{20}, 310)$       | $(2^{17}, 430)$             | $(2^{-2}, 230, 0.35)$                    | $(2^{-5}, 270, 0.35)$                      | $(2^3, 2^{23})$             | $(2^7, 2^{-10})$               |
| Breast cancer     | $(2^{-3}, 80)$        | $(2^{-1}, 160)$             | $(2^{-6}, 90, 0.15)$                     | $(2^{13}, 290, 0.25)$                      | $(2^6, 2^1)$                | $(2^6, 2^{17})$                |
| Balloon           | $(2^{22}, 160)$       | $(2^{16}, 190)$             | $(2^{13}, 250, 0.2)$                     | $(2^{17}, 210, 0.2)$                       | $(2^1, 2^8)$                | $(2^0, 2^{-7})$                |
| Auto-MPG          | $(2^7, 630)$          | $(2^{-10}, 820)$            | $(2^{21}, 380, 0.35)$                    | $(2^{11}, 680, 0.3)$                       | $(2^{-2}, 2^{-1})$          | $(2^1, 2^5)$                   |
| Bank              | $(2^{-1}, 660)$       | $(2^0, 590)$                | $(2^2, 710, 0.3)$                        | $(2^{-9}, 450, 0.15)$                      | $(2^{10}, 2^{-4})$          | $(2^{21}, 2^{-5})$             |
| Census (house8L)  | $(2^3, 580)$          | $(2^5, 460)$                | $(2^1, 190, 0.3)$                        | $(2^3, 600, 0.25)$                         | $(2^5, 2^{-6})$             | $(2^0, 2^{-4})$                |

and least-square support vector regression [37]. The results comparisons of RAE-ELM and other learning algorithms for real-world data regressions are shown in Table 4.

Table 4 lists the averaging results of multiple trails of the four ELM based algorithms (RAE-ELM, basic ELM, original Ada-ELM [30], and modified Ada-ELM [31]), SVR [36], and LS-SVR [37] for fourteen representative real-world data regression problems. The selected datasets include large scale of data and small scale of data, as well as high dimensional data problems and low dimensional problems. It is easy to find that averaged testing RMSE obtained by RAE-ELM for all the fourteen cases are always the best among these six algorithms. For original Ada-ELM, the performance is sensitive to the selection of threshold value of  $\Phi$ . The best performed original Ada-ELM models for different regression problems own their correspondingly different threshold values. Therefore, the manual chosen strategy is not good. The generalization performance of modified Ada-ELM, in general, is better than original Ada-ELM. However, the empirical suggested initial threshold value at 0.2 does not ensure a mapping to the best performed regression model.

The three AdaBoost.RT based ensemble ELMs (RAE-ELM, original Ada-ELM, and modified Ada-ELM) all perform better than the basic ELM, which verifies that an ensemble ELM using AdaBoost.RT can achieve better prediction accuracy than using individual ELM as the predictor. The averaged generalization performance of basic ELM is better than SVR while it is slightly worse than LS-SVR.

To find the best performed original Ada-ELM model or modified Ada-ELM for a regression problem, as the input dataset and ELM networks are not related to the threshold selection, the optimal parameter need be searched by brute-force. One needs to carry out a set of experiments with different (initial) threshold values and then searches among them for the best ensemble ELM model. Such process is time consuming. Moreover, the generalization performance of the optimized ensemble ELMs using original Ada-ELM or modified Ada-ELM can hardly be better than that of the proposed RAE-ELM. In fact, the proposed RAE-ELM is always the best performed learner among the six candidates for all fourteen real-world regression problems.

TABLE 4: Result comparisons of testing RMSE of RAE-ELM and other learning algorithms for real-world data regression problems.

| Datasets          | RAE-ELM       | Basic ELM [4] | Original AdaELM [30] | Modified AdaELM [31] | SVR [36] | LSSVR [37] |
|-------------------|---------------|---------------|----------------------|----------------------|----------|------------|
| LVST              | <b>0.2571</b> | 0.2854        | 0.2702               | 0.2653               | 0.2849   | 0.2801     |
| Yeast             | <b>0.1071</b> | 0.1224        | 0.1195               | 0.1156               | 0.1238   | 0.1182     |
| Computer hardware | <b>0.0563</b> | 0.0839        | 0.0821               | 0.0726               | 0.0976   | 0.0771     |
| Abalone           | <b>0.0731</b> | 0.0882        | 0.0793               | 0.0778               | 0.0890   | 0.0815     |
| Servo             | <b>0.0727</b> | 0.0924        | 0.0884               | 0.0839               | 0.0966   | 0.0870     |
| Parkinson disease | <b>0.2219</b> | 0.2549        | 0.2513               | 0.2383               | 0.2547   | 0.2437     |
| Housing           | <b>0.1018</b> | 0.1259        | 0.1163               | 0.1135               | 0.127    | 0.1185     |
| Cloud             | <b>0.2897</b> | 0.3269        | 0.3118               | 0.3006               | 0.3316   | 0.3177     |
| Auto price        | <b>0.0809</b> | 0.1036        | 0.0910               | 0.0894               | 0.1045   | 0.0973     |
| Breast cancer     | <b>0.2463</b> | 0.2641        | 0.2601               | 0.2519               | 0.2657   | 0.2597     |
| Balloon           | <b>0.0570</b> | 0.0639        | 0.0611               | 0.0592               | 0.0672   | 0.0618     |
| Auto-MPG          | <b>0.0724</b> | 0.0862        | 0.0799               | 0.0781               | 0.0891   | 0.0857     |
| Bank              | <b>0.0415</b> | 0.0551        | 0.0496               | 0.0453               | 0.0537   | 0.0498     |
| Census (house8L)  | <b>0.0746</b> | 0.0820        | 0.0802               | 0.0795               | 0.0867   | 0.0813     |

## 6. Conclusion

In this paper, a robust AdaBoost.RT based ensemble ELM (RAE-ELM) for regression problems is proposed, which combined ELM with the novel robust AdaBoost.RT algorithm. Combining the effective learner, ELM, with the novel ensemble method, the robust AdaBoost.RT algorithm could construct a hybrid method that inherits their intrinsic properties and achieves better prediction accuracy than using only individual ELM as predictor. ELM tends to reach the solutions straightforwardly, and the error rate of regression prediction is, in general, much smaller than 0.5. Therefore, selecting ELM as the “weak” learner can avoid overfitting. Moreover, as ELM is a fast learner with quite high regression performance, it contributes to the overall generalization performance of the ensemble ELM.

The proposed robust AdaBoost.RT algorithm overcomes the limitations existing in the available AdaBoost.RT algorithm and its variants where the threshold value is manually specified, which may only be ideal for a very limited set of cases. The new robust AdaBoost.RT algorithm is proposed to utilize the statistics distribution of approximation error to dynamically determine a robust threshold. The robust threshold for each weak learner  $WL_t$  is self-adjustable and is defined as the scaled standard deviation of the approximation errors,  $\lambda\sigma_t$ . We analyze the convergence of the proposed robust AdaBoost.RT algorithm. It has been proved that the error of the final hypothesis output by the proposed ensemble algorithm,  $E_{\text{ensemble}}$ , is within a significantly superior bound.

The proposed RAE-ELM is robust with respect to the difference in various regression problems and variation of approximation error rates that do not significantly affect its highly stable generalization performance. As one of the key parameters in ensemble algorithm, threshold value does not need any human intervention; instead, it is able to be self-adjusted according to the real regression effect of ELM networks on the input dataset. Such mechanism enable RAE-ELM to make sensitive and adaptive adjustment to the intrinsic properties of the given regression problem.

The experimental result comparisons in terms of stability and accuracy among the six prevailing algorithms (RAE-ELM, basic ELM, original Ada-ELM, modified Ada-ELM, SVR, and LS-SVR) for regression issues verify that all the AdaBoost.RT based ensemble ELMs perform better than the SVR, and, more remarkably, the proposed RAE-ELM always achieves the best performance. The boosting effect of the proposed method is not significant for small sized and low dimensional problems as the individual classifier (ELM network) could already be sufficient to handle such problems well. It is worth pointing out that the proposed RAE-ELM has better performance than others especially for high dimensional or large sized datasets, which is a convincing indicator for good generalization performance.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors would like to thank Professor Guang-Bin Huang from Nanyang Technological University, for providing inspiring comments and suggestions on our research. This work is financially supported by the University of Macau with Grant no. MYRG079(Y1-L2)-FST13-YZX.

## References

- [1] C. L. Philip Chen, J. Wang, C. H. Wang, and L. Chen, “A new learning algorithm for a Fully Connected Fuzzy Inference System (F-CONFIS),” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1741–1757, 2014.
- [2] Z. Yang, P. K. Wong, C. M. Vong, J. Zhong, and J. Y. Liang, “Simultaneous-fault diagnosis of gas turbine generator systems using a pairwise-coupled probabilistic classifier,” *Mathematical Problems in Engineering*, vol. 2013, Article ID 827128, 13 pages, 2013.

- [3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.
- [4] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513-529, 2012.
- [5] S. C. Ng, C. C. Cheung, and S. H. Leung, "Magnified gradient function with deterministic weight modification in adaptive learning," *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1411-1423, 2004.
- [6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.
- [7] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 4, pp. 1067-1072, 2009.
- [8] J. Cao, Z. Lin, and G. B. Huang, "Voting base online sequential extreme learning machine for multi-class classification," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '13)*, pp. 2327-2330, IEEE, 2013.
- [9] J. Cao, Z. Lin, G. B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66-77, 2012.
- [10] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.
- [11] J. Luo, C. M. Vong, and P. K. Wong, "Sparse Bayesian extreme learning machine for multi-classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 836-843, 2014.
- [12] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, vol. 2, MIT Press, Cambridge, UK, 2006.
- [13] Y. Lan, Y. C. Soh, and G. B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13, pp. 3391-3395, 2009.
- [14] N. Liu and H. Wang, "Ensemble based extreme learning machine," *IEEE Signal Processing Letters*, vol. 17, no. 8, pp. 754-757, 2010.
- [15] X. Xue, M. Yao, Z. Wu, and J. Yang, "Genetic ensemble of extreme learning machine," *Neurocomputing*, vol. 129, no. 10, pp. 175-184, 2014.
- [16] X. L. Wang, Y. Y. Chen, H. Zhao, and B. L. Lu, "Parallelized extreme learning machine ensemble based on min-max modular network," *Neurocomputing*, vol. 128, pp. 31-41, 2014.
- [17] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708-713, 1979.
- [18] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.
- [19] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.
- [20] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197-227, 1990.
- [21] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms/Robert E. Schapire, Yoav Freund*, MIT Press, Cambridge, Mass, USA, 2012.
- [22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
- [23] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79-87, 1991.
- [24] V. Guruswami and A. Sahai, "Multiclass learning, boosting, and error-correcting codes," in *Proceedings of the 12th Annual Conference on Computational Learning Theory (COLT '99)*, pp. 145-155, July 1999.
- [25] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297-336, 1999.
- [26] N. Li, "Multiclass boosting with repartitioning," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 569-576, ACM, June 2006.
- [27] H. Drucker, "Improving regressors using boosting," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 107-115, 1997.
- [28] R. Avnimelech and N. Intrator, "Boosting regression estimators," *Neural Computation*, vol. 11, no. 2, pp. 499-520, 1999.
- [29] D. P. Solomatine and D. L. Shrestha, "AdaBoost.RT: a boosting algorithm for regression problems," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 1163-1168, IEEE, 2004.
- [30] D. L. Shrestha and D. P. Solomatine, "Experiments with AdaBoost.RT, an improved boosting scheme for regression," *Neural Computation*, vol. 18, no. 7, pp. 1678-1710, 2006.
- [31] H.-X. Tian and Z.-Z. Mao, "An ensemble ELM based on modified AdaBoost.RT algorithm for predicting the temperature of molten steel in ladle furnace," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 73-80, 2010.
- [32] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, Hangzhou, China, June 2014.
- [33] J. Cao, Z. Lin, and G.-B. Huang, "Composite function wavelet neural networks with extreme learning machine," *Neurocomputing*, vol. 73, no. 7-9, pp. 1405-1416, 2010.
- [34] R. Feely, *Predicting stock market volatility using neural networks*, B.A. (Mod.) [Ph.D. thesis], Trinity College Dublin, Dublin, Ireland, 2000.
- [35] K. Bache and M. Lichman, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, Calif, USA, 2014, <http://archive.ics.uci.edu/ml>.
- [36] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in Neural Information Processing Systems*, vol. 9, pp. 155-161, 1997.
- [37] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293-300, 1999.

## Research Article

# Weibo Information Propagation Dissemination Based on User Behavior Using ELM

**Huilin Liu and Yao Li**

*College of Information Science and Engineering, Northeastern University, Shenyang, Liaoning 110819, China*

Correspondence should be addressed to Huilin Liu; [liuhuilin@mail.neu.edu.cn](mailto:liuhuilin@mail.neu.edu.cn)

Received 21 September 2014; Revised 1 January 2015; Accepted 16 January 2015

Academic Editor: Tao Chen

Copyright © 2015 H. Liu and Y. Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Information dissemination prediction based on Weibo has been a hot topic in recent years. In order to study this, people always extract features and use machine learning algorithms to do the prediction. But there are some disadvantages. Aiming at these deficiencies, we proposed a new feature, the dependency between the Weibos involved in geographical locations and location of the user. We use ELM to predict behaviors of users. An information dissemination prediction model has also been proposed in this paper. Experimental results show that our proposed new feature is real and effective, and the model we proposed can accurately predict the scale of information dissemination. It also can be seen in the experimental results that the use of ELM significantly reduces the time, and it has a better performance than the traditional method based on SVM.

## 1. Introduction

With the development of the web 2.0, social networks have become an indispensable part of people's lives. Large social networking site like Facebook, Twitter, and so forth brings a lot of happy time to people. Sina Weibo, as one of China's largest online social networks, has more than 500 million registered users. Every day these users produce a lot of social network data through continuously released and forwarded microblogging. These social network data researches help enterprises and government find the users network behavior rules and make the corresponding measures. Thus, the study of Weibo is a hot issue in recent years.

There are a lot of directions on the study of Weibo, including sentiment analysis based on Weibo [1] and Weibo personalized recommendation research [2]. One high practical value direction of the researches in Weibo is studying online behavior of users and corresponding information propagation. This aspect of the study can help enterprises to understand the user behavior mode, grasp the user interest preference and recommend the interest topics, other users, and groups to the user. It can also help the government to understand the range of the spread of news, judge social public opinion direction and reactions, and adjust corresponding policies in time.

There are many researches about user behavior in online social networks and information dissemination exists. One of the common methods is extracting user behavior characteristics and use machine learning algorithm to classify and predict user behavior [3–6]. In general, the researchers adopt support vector machine (SVM) algorithm. The features they widely use are the influence of user, the intimacy between the users, the interest similarity of user, Weibo content importance, and so forth.

In life, people are more concerned about the information around their side. This can also be extended to Weibo. So if a Weibo involves the geographical location, the users who are near the location will pay more attention to the Weibo than users in other areas. Although there are a lot of social network applications which use the geographical position, for example, Lingad et al. [7] studied the extraction of Weibo position related to the disaster, Hosseini et al. [8] studied location oriented phrase detection in microblogs. But on the analysis of user online behavior and information dissemination, the dependency between the geographical locations in which Weibos are involved and location of the user has not been mentioned.

Therefore, on the basis of summarizing the work before, we take the dependency between the geographical locations in which Weibos involved and location of the user as a new

feature to analyze user behavior and information dissemination. At the same time, because extreme learning machine algorithm runs fast and can get the optimal solution rather than the sub-optimal solutions, we adopt ELM to replace SVM. The main contribution of this paper is shown as below.

- (1) We propose a new feature, the dependency between the geographical locations in which Weibos involved and location of the user. We use this feature and other proposed feature to analyze user behavior and information dissemination.
- (2) We test the different performance between the different value of  $\Delta t$  in ignore dataset and found that when  $\Delta t$  is 30 minutes, the performance is the best.
- (3) We use ELM instead of SVM to predict user behavior and information dissemination.

Our experimental results show that, with the new feature we proposed, we get a higher forecasting accuracy than without the new feature. Our experimental results also show that ELM gets higher accuracy than SVM in the same dataset.

The rest of this paper is organized as follows. Section 2 briefly introduces the related work about online social network and ELM. Section 3 introduces the data and feature we use to predict user behavior and the information dissemination model. And the experimental results are reported in Section 4. Finally, we present our conclusions and future work in Section 5.

## 2. Related Work

*2.1. Online Social Network.* Due to the popularity of social networks, there are many studies of social networks. For example, Marques and Serrão [9] proposed using rights management systems to improve the content privacy of social network users; Quang et al. [10] found the cluster of actors in social network based on the topic of messages; Tseng and Chen [11] proposed incremental SVM model to detect unwanted email, and so on.

Our main work in this paper is analyzing user behavior and information dissemination. There are a lot of related works of this aspect. Song et al. [3] proposed 4 features to predict if user will forward the Weibo or ignore it. The features are the authority of user, the activity of user, the preference of user, and the social relations of user. The four features can reflect the user behavior to a certain extent, but they did not consider the importance of Weibo content and the dependency between the geographical locations, which are involved Weibos, and locations of the user. Zaman et al. adopted the model of collaborative filtering based on probability [12, 13]. They select the user name, the number of attention, and number of words that Weibo contains to predict the forward behavior of user. Although these features have some influence on user behavior and information dissemination, these features are not the main factor affecting the user's behavior. Cao et al. [4] improved the prediction model, added the Weibo content length, Weibo importance, whether the user is authenticated user, and some other features. The added features improved the prediction

accuracy of user behavior and information dissemination, but they still did not consider the relationship between Weibo mention place names and users.

Some other works also give us some help. For example, some people analyzed the flow of information within the scope of the blog and made a prediction model of information transmission in [6]. Sina Weibo and the traditional blog have certain similarities. We can draw lessons from the spread of the blog. Webberley et al. [14] studied the transmit delay, the depth and breadth of information dissemination on Twitter. They preliminary studied user behavior patterns and forwarding rules and have certain reference significance.

Some researchers have studied the influence of mentioned location on information dissemination. For example, Bandari et al. [15] put forward an algorithm to predict whether the news is popular enough on Twitter or whether it can trigger a heated discussion on social networking sites. This paper puts forward four features: article categories, the degree of objective, the article mentioned geographical name and people name, and the sources of article. But the study only gives the effect of the popular places to information dissemination, does not take the dependency between the geographical name and users into account.

In conclusion, we propose a new feature: the dependency between the geographical locations, which are involved Weibos, and locations of the user.

*2.2. ELM.* Extreme Learning Machine (ELM) is put forward by Huang at Nanyang technological university in 2004 [16]. It is a more simple and effective algorithm of single hidden layer feed forward network (SLFNs) algorithm. It can automatically choose the input weight and analyze decision output weight. It provides the best generalization ability and very fast learning speed. Huang has proved in Extreme Learning Machine a New Learning Scheme of Feed forward Neural Networks, that under the same condition of the classification, ELM rate is much higher than the SVM. According to Professor Huang previous studies [17, 18], we summarize the ELM theory is as follows.

For  $n$  different samples  $(x_i, t_i)$ ,  $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T \in R^n$  and  $t_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^m$ . If the SLFNs has  $\tilde{N}$  hidden nodes and its activation function is  $g(x)$ , then we get the formula as follows:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(w_i \cdot x_j + b_i) = o_j, \quad (1)$$

$$j = 1, 2, \dots, N.$$

In the formula,  $w_i = [w_{i1}, w_{i2}, \dots, w_{im}]^T$  is the weight vector which connects  $i$ th hidden node with the input vector.  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the weight vector which connects  $i$ th hidden node with the output vector.  $b_i$  is the threshold of the  $i$ th hidden node.  $w_i \cdot x_j$  is the inner product of  $w_i$  and  $x_j$ .

The  $N$  samples approximate to zero mean error, so we have  $\sum_{j=1}^{\bar{N}} \|o_j - t_j\| = 0$ ; then, we get the formula as follows:

$$\sum_{i=1}^{\bar{N}} \beta_i g(w_i \cdot x_j + b_i) = t_j, \quad j = 1, 2, \dots, N. \quad (2)$$

The above formula can be written into  $H\beta = T$ . Then, the process of ELM can be mathematically modeled as the following formula:

$$\text{Minimize: } \|H\beta - T\|^2, \|\beta\|. \quad (3)$$

Here,  $H$  can be expressed as

$$H(w_1, \dots, w_{\bar{N}}, b_1, \dots, b_{\bar{N}}, x_1, \dots, x_{\bar{N}}) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{\bar{N}} \cdot x_1 + b_{\bar{N}}) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{\bar{N}} \cdot x_N + b_{\bar{N}}) \end{bmatrix}_{N \times \bar{N}}. \quad (4)$$

Therefore, we get a solution for the parameter  $i$  as

$$\hat{\beta} = H^\dagger T, \quad (5)$$

where  $H^\dagger$  is the Moore-Penrose generalized inverse of matrix  $H$ .

Based on the above analysis, the machine learning-based algorithm without iterative tuning can be divided into three steps. The specific process of ELM is summarized as follows.

*Step 1.* Randomly assign input weight  $w_i$  and bias  $b_i$ ,  $i = 1, 2, \dots, \bar{N}$ .

*Step 2.* Calculate the hidden layer output matrix  $H$ .

*Step 3.* Calculate the output weight  $\beta$ , where  $\beta = H^\dagger T$ .

Compared with SVM, ELM can be directly applied in many kinds of classification problems. In professor Huang Extreme Learning Machine for Regression and Multiclass Classification study, he has proved that the SVM obtains sub-optimal solution and needs higher computational complexity [19]. Therefore, ELM has the advantages that SVM does not have and has a broad application prospect.

### 3. User Behavior and Information Dissemination Prediction

In this paper, we analyze people's behavior and information dissemination on Weibo. First of all, we need to get the data from Sina Weibo. The behaviors of users in Sina Weibo are releasing, browsing, commenting, and forwarding. Release and forward behaviors are associated with information dissemination. However, the release behavior is decided by users self and we cannot control it. So our main study is forward behavior of users.

In this section, we will introduce the data and features we use and give the information dissemination prediction model we proposed. First of all, we give the dataset description.

*3.1. Dataset Description.* When we get the Sina Weibo data, first of all, we choose one user and get its fans list. Second according to the fans list, we get the fans list of each user in fans list. In this method, finally we get a user's dataset. We got 96438 users in this dataset. Sina Weibo users can be roughly divided into three categories: release active users, forward active users, and inactive users. If a user does not have forward or release activity in 1 month, we think it is an inactive user. Because the inactive users do not have any contribution to the user behavior and information dissemination prediction, so we excluded these users. Finally we got 89377 users in the dataset. Then, we crawl all Weibos of these users which published between May 1, 2014, and May 31, 2014, and get 564835 Weibos. In these Weibos, there are 114943 Weibos related to geographical locations. Most of the Sina Weibos are Chinese Weibos, the geographical locations in them are Chinese location. So the small amount of Weibos which contain foreign geographical locations are consider to have nothing to do with the geographical location. We select the data from the whole Weibo dataset to build forward and ignore datasets. Because we cannot see the ignore behavior directly, we need to define the ignore dataset first. The definition of ignore dataset shown as follows.

*Definition 1* (ignore dataset). If user  $u$  forwarded the Weibo published at time  $t$ , the Weibos which published by the friends of the user at  $[t - \Delta t, t + \Delta t]$  and are not forwarded by the user are the ignore samples. All the ignore sample constitute ignore dataset.

Users ignore the Weibos not only because users do not like them, but also because they are leaving and not seeing the Weibos. So we selected 10 minutes, 30 minutes, 1 hour, 2 hours, and 12 hours as  $\Delta t$ . We also studied influence of different ignore datasets to the final accuracy. Algorithm 1 is used to find ignore dataset.

In order to facilitate our location keywords extraction, we established the province tree to identify the place name. Figure 1 is the structure of the province tree.

As we can see in Figure 1, China, according to the position, is divided into east China, south China, central China, north China, northwest, southwest, and northeast. Each region contains some provinces, and each province contains a number of cities. According to the province tree, we can identify the key word belonging to which geographical locations. We can also get the subordinate situation of the key word.

In province tree, we only consider the city name, without regard to the block name. This is because, in China, different city may contain the same blocks name. We cannot be able to accurately determine the block belongs to which city.

Our study is based on the above data. In the next section, we will introduce the features we use and the corresponding evaluation index.

*3.2. Feature Description.* In this section, we will introduce the features we use. First of all, we will introduce the new feature we proposed. And then we will introduce other features we use.

Inputs: Weibos set  $P$  which published by the friends of the user  $u$ ;  
 Weibos set  $Q$  which user  $u$  forward.  
 Output: Weibos set  $R$  which user  $u$  ignore.  
 (1) Any Weibo  $m, m \in Q$ , read the publish time  $t_m$ ;  
 (2) Find Weibo  $w, w \in P$   
 (3) while (the publish time of  $w$  satisfy  $t \in [t_m - \Delta t, t_m + \Delta t]$ )  
 (4)  $w \in S$ ; //  $S$  is an intermediate variable.  
 (5) While ( $\forall w, w \in S, w \notin Q$ )  
 (6) Add  $w$  to  $R$ ;  
 (7) Output  $R$ ;

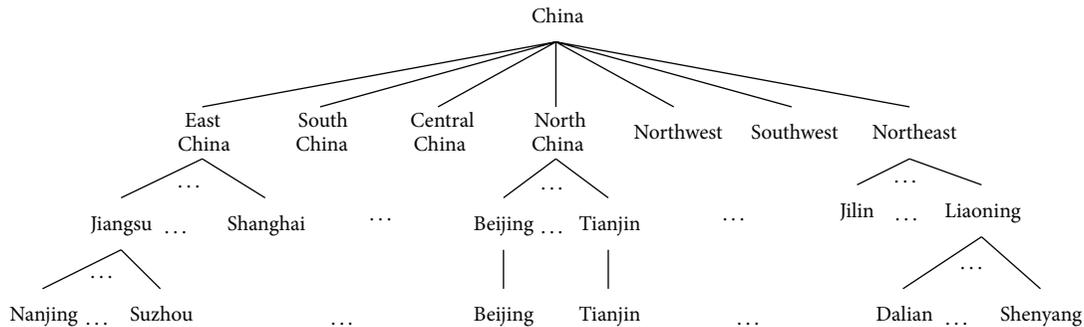
ALGORITHM 1: Ignore( $P, Q$ ).

FIGURE 1: The structure of the province tree.

3.2.1. *The Dependency between the Weibos Involved Geographical Locations and Location of the User.* The Weibo involved geographical locations have been proposed before. However, they only concern whether the location name is famous and do not connect it with the locations of users. As the government starts carrying out internet political communication on Weibo, this connection becomes more and more important. Information published by the local government is likely to be paid attention to in the local and surrounding areas. The further area users will give less attention to it. We use Peking University PKUVIS Weibo visual analysis tools [20] to analyze 150 Weibos and one of it is shown as follows:

#毛絮是虫子# 【🐛南京满天飞的“毛絮”竟是长着白毛的虫子!】@现代快报:这两天,南京一些地方飘着柳絮一样的东西,漫天飞舞。南京林业大学森环院的专家发现,其实它们根本不是柳絮,而是活物小虫子!!叫“榆四脉绵蚜”!今年的气候有利于它们的繁殖,所以数量非常多!🐛我整个人都不好了!

In this Weibo, we can extract the location name Nanjing. According to the province tree, it belongs to Jiangsu province. We guess the users in Jiangsu may have high attention in this Weibo. The users far from Jiangsu may pay less attention. So we count users number in every province who forward this Weibo. According to the province field of the data, we obtained the province of these Weibos users. Sina Weibo use code to represent the provinces and cities. Table 1 shows the provinces and its corresponding code. For convenience, in the following figure, we all use the province codes in Table 2

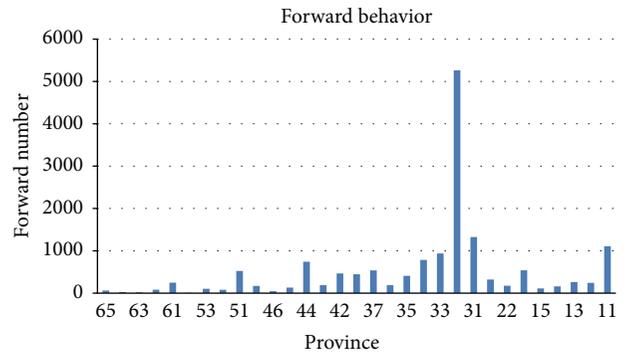


FIGURE 2: The number of users in every province.

to represent the province. Figure 2 shows the number of users in every province.

We can see in Figure 2, the local users in Jiangsu pay the most attention to the Weibo. The locations which near Jiangsu also pay much attention to it (like Anhui, Shanghai, Zhejiang, and Shandong).

According to the theory of probability, to other provinces and cities, the forwarding quantity percentage should have the same regularity with the registered users' percentage in each province. It is hard to get the registered users' percentage. But in Figure 1 we can see the economically developed provinces, such as Beijing and Guangzhou, have higher forward number than some underdeveloped areas like

TABLE 1: Province and its code.

|           |              |          |          |          |                |          |         |
|-----------|--------------|----------|----------|----------|----------------|----------|---------|
| Provinces | Beijing      | Tianjin  | Hebei    | Shanxi   | Inner Mongolia | Liaoning | Jilin   |
| Code      | 11           | 12       | 13       | 14       | 15             | 21       | 22      |
| Provinces | Heilongjiang | Shanghai | Jiangsu  | Zhejiang | Anhui          | Fujian   | Jiangxi |
| Code      | 23           | 31       | 32       | 33       | 34             | 35       | 36      |
| Provinces | Shandong     | Henan    | Hubei    | Hunan    | Guangdong      | Guangxi  | Hainan  |
| Code      | 37           | 41       | 42       | 43       | 44             | 45       | 46      |
| Provinces | Chongqing    | Sichuan  | Guizhou  | Yunnan   | Tibet          | Shaanxi  | Gansu   |
| Code      | 50           | 51       | 52       | 53       | 54             | 61       | 62      |
| Provinces | Qinghai      | Ningxia  | Sinkiang |          |                |          |         |
| Code      | 63           | 64       | 65       |          |                |          |         |

TABLE 2: Ignore samples.

| $\Delta t$ | Quantity |
|------------|----------|
| 15 minutes | 72996    |
| 30 minutes | 119392   |
| 1 hour     | 188376   |
| 2 hours    | 307483   |
| 12 hours   | 431645   |

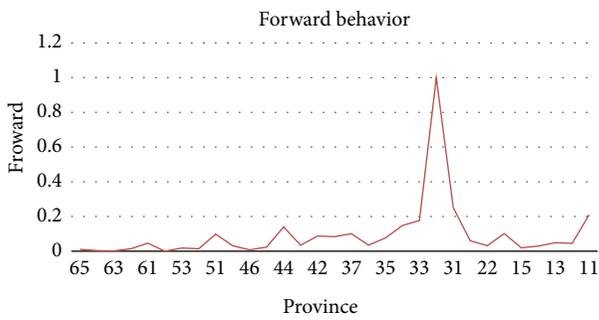


FIGURE 3: The normalization of forwarding number.

Sinkiang and Ningxia. We guess this is because people in developed cities occupy more network resources and can easily get the website, so the users in developed cities may be larger than underdeveloped city. The other Weibos also have this rule.

To represent the cities' development, we found the per capita GDP in each province in 2013. Forward number and per capita GDP are not in the same magnitude. So we normalized these data. Figure 3 shows the normalized forward number. Figure 4 shows the normalized per capita GDP.

In Figures 3 and 4 we can see, in addition to geographical location mentioned in the Weibo, the forward quantity and the per capita GDP in other province are in the same regularity. For example, in Beijing, Guangdong and other regions, two figures both have a local peak. The geographical location mentioned in the Weibo makes this feature not obvious. This further proves that the geographical location mentioned in the Weibo has a stronger influence on the users who are close to it.

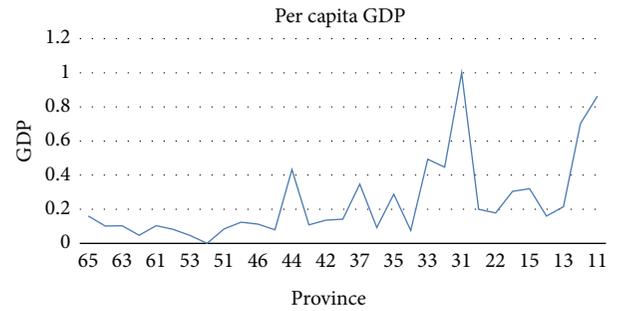


FIGURE 4: The per capita GDP in 2013.

All the Weibos we tested have this conclusion. So we use the per capita GDP to represent the registered users' percentage. And then the per capita GDP can represent the possibility of users forwarding. When the province is the geographical locations involved Weibos, we add 0.5 to the per capita GDP, which means this geographical location plays a predominant role in the forwarding behavior. The final value represents the dependency between the Weibos involved geographical locations and locations of the users.

Besides the new feature we put forward, other features are widely applied to the user behavior analysis and Information Propagation Dissemination. Researchers in [3] selected 4 features to judge user forward behavior. The features are The User's Authority, User's Activity, User's Preference, and User's Social Relations. However, the user's authority is relevant to the user's forward behavior, but the correlation is weak. Researchers in [4] selected 15 features. But some features are covered by other features. For example, when we compute the PageRank, the user fan numbers are used. This kind of features is useless and should not be used in the user forward behavior prediction.

Another research RT to Win! Predicting Message Propagation in Twitter [21] divided features into two categories. There are 7 social features (i.e., number of followers, friends, statuses, favorites, number of times the user was listed, is the user verified, is the user's language English) and 7 tweet features (i.e., number of followers, friends, statuses, favorites, number of times the user was listed, is the user verified, is the user's language English).

To summarize the features in the above and other papers, we selected 5 features to forecast user forwarding behavior. They are the influence of user, user release activity, and forward activity, the intimacy between the users, the interest similarity between user and content or between users and Weibo content importance. The following is these features in detail.

**3.2.2. The Influence of User.** People always use PageRank to compute the influence of user [22]. The PageRank algorithm is used to measure the importance of specific pages relative to other pages in the search engine. The PageRank formula they use is shown as

$$pr_i = \frac{1-q}{N} + q \sum_{j \in \text{Follower}(i)} \frac{pr_j}{|\text{Friend}(j)|}. \quad (6)$$

In this formula,  $pr_i$  represent the PageRank value of user  $I$ ,  $\text{Follower}(i)$  represents the fans list of user  $I$ ,  $\text{Friend}(j)$  represents the collection of users that user  $j$  pays attention to,  $q$  is the damping coefficient, and  $N$  is the total number of users.

**3.2.3. User Release Activity and Forward Activity.** Because of the different behaviors of the user, the user activity can be divided into two aspects, the user release activity and forward activity. The user release activity is the Weibo number published over a period of time. We can use formula (7) to compute it:

$$PA = \frac{n}{t}. \quad (7)$$

The  $PA$  in formula (7) represents the Weibo number published over a period of time,  $n$  is the total number of Weibo,  $t$  is the unit time. In general, we set  $t$  to 1 day.

The forward activity is percentage of users forwarding Weibo account for all published Weibo in one day. We use formula (8) to compute it:

$$RA = \frac{\sum_{i \in t} r_i}{\sum_{i \in t} p_i}. \quad (8)$$

$r_i$  is the number of users forwarding Weibo in  $i$ th day,  $p_i$  is the number of users releasing Weibo in  $i$ th day, and  $RA$  represents the forward activity. The higher the  $RA$  is, the more active the users are. Users with high forward frequency play a bigger role in information dissemination.

**3.2.4. The Intimacy between the Users.** Because the forward behavior in Weibo can reflect the interaction between the users better, we compute the intimacy between the users by calculating the percentage of Weibo published by the upstream user in the forwarding Weibo of the user. The formula we use is

$$f_{uv} = \frac{n_{uv}}{n_u}. \quad (9)$$

In this formula,  $n_{uv}$  represents the number of the Weibos of user  $v$  which appears in the forward Weibo of user  $u$ .  $n_u$  represents the total number of forward Weibo of user  $u$ .

**3.2.5. The Interest Similarity between User and Content or between Users.** Weibo can reflect the interests of users. The larger the interest similarity between user and content, the greater the chance user forward. The larger the interest similarity between user and upstream user, the greater the chance user forward. So we need to compute the interest similarity. Because the user's interest is the change over time, we need to analyze the Weibo which release time near a few days. Interest space is extracted from weibo, and the following is the process of compare.

- (1) Collect user interest. We select a user and collect the user  $m$  Weibo published nearly five days. These form the user interest space  $I_s = \{s_i, 0 < i < m\}$ .  $I_s$  is the interest space of user  $s$  and  $s_i$  is the  $i$ th Weibo of user  $s$ .
- (2) Participle. For Weibo in Chinese, we use the Chinese Academy of Sciences Chinese lexical analysis system ICTCLAS do the word segmentation [22]. For Weibo in English, we use space. We get the words level interest space  $I_\omega = \{\omega_i\}$ .  $\omega_i$  is the  $i$ th word.
- (3) Remove the stop. We remove the stop word and get the new words level interest space  $I = \{\omega_i\}$ .
- (4) Repeat (2) and (3); we get the interest space of user  $s$   $J_s = \{\omega_j\}$ .
- (5) For the two users  $s$  and  $s'$ , we calculate the similarity of  $J_s$  and  $J_{s'}$ . For the user  $s$  and the content  $t$ , we calculate the similarity of  $J_s$  and  $J_t$ . We use Jaccard formula to calculate the similarity [23]. The Jaccard formula is

$$\text{Jaccard}(J_s, J_{s'}) = \frac{J_s \cap J_{s'}}{J_s \cup J_{s'}}. \quad (10)$$

**3.2.6. Weibo Content Importance.** Usually if a Weibo contains significant events or popular information, the forward rate will be high. So the importance of Weibo content can help us analyze Weibo information dissemination. Based on computing weight of TF-IDF (term frequency inversed document frequency) algorithm on the text classification field, we calculate the importance of Weibo [24]. The thought of this algorithm is that in a specific document the higher the frequency of word appears in the document, the more important the word is; the lower the frequency of word appears in other document, the more important the word is. We can use formula (11) to calculate the importance:

$$tf(d) = n_\omega \times \log \frac{N}{n_d}. \quad (11)$$

In this formula,  $d$  represents the word  $d$  in the Weibo  $\omega$ ,  $n_\omega$  represents the number of  $d$  appearing in  $\omega$ ,  $N$  represents the number of Weibo that Weibo set  $W$  contains, and  $n_d$  represents the number of Weibos containing  $d$  in the Weibo set  $W$ . The TF-IDF of Weibo  $\omega$  can be computed by adding the TF-IDF of all the word in  $\omega$ :

$$tf(\omega) = \sum_j tf(d_j). \quad (12)$$

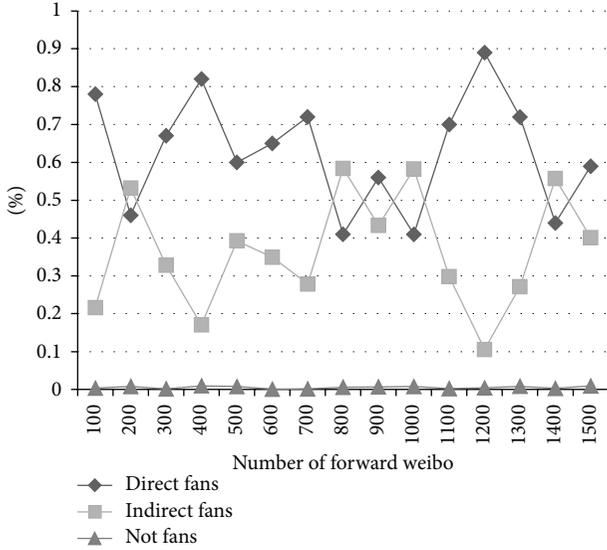


FIGURE 5: Each percentage of 3 forward behaviors.

**3.3. Information Dissemination Prediction Model.** According to the features in Section 3.2, we use ELM to forecast the user forward behavior. According to the predicted forward behavior, we forecast information dissemination scale.

The forward behaviors in Weibo can be divided into 3 aspects: direct fans forwarding, indirect fans forwarding, and not fans forwarding. We count the each percentage of 3 forward behaviors in different scales of Weibo. Figure 5 shows each percentage of 3 forward behaviors from the size of 100 to the size of 1500.

We can see from Figure 5 that forward behaviors are mainly composed of direct fans and indirect fans. The percentage of not fan users is almost 0. So we ignore the forward behaviors of not fan users.

When we make the prediction, we start from Weibo publishers. And then we traverse its list of fans and predict if the fan will forward the Weibo. If the fan forwards it, the forwarded number increases 1. Then, traverse the fans list of this user. We repeat iteration like this until no users forward the Weibo. The prediction model can be represented by a tree. Figure 6 is a simple example of prediction model tree.

The gray point in Figure 6 is the publisher of Weibo. The black points are the users who will forward the Weibo. The white points are the users who will not forward the Weibo. When we make the prediction, we start from the user  $U_0$  and traverse its fans list. We find the fans list contains 3 users:  $U_1$ ,  $U_6$ , and  $U_7$ , and the  $U_1$  is the forwarding point. Thus, the forwarded number increases 1 and we traverse the fans list of  $U_1$ . The fans list of  $U_1$  contains 2 points,  $U_2$  and  $U_3$ .  $U_2$  is not the forwarding point, but  $U_3$  is the forwarding point. So the forwarded number increases 1 and we traverse the fans list of  $U_3$ . The points in fans list of  $U_3$  are  $U_4$  and  $U_5$ , and both of them are not the forwarding points. So we come to  $U_6$ . The handling of  $U_6$  is similar to the above, and in this method, we finally got all the forwarding nodes.

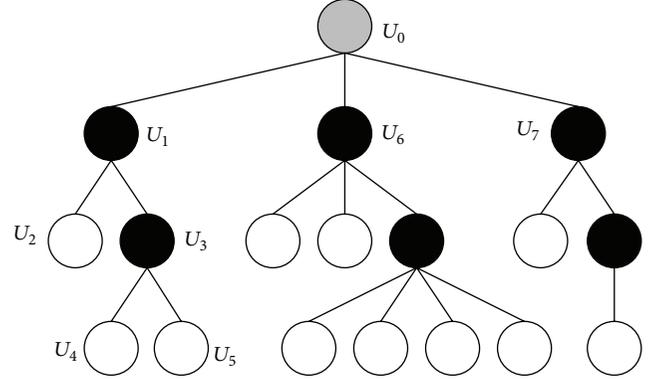


FIGURE 6: Prediction model tree.

TABLE 3: User forward behavior prediction results using ELM.

| $\Delta t$ | Accuracy | Recall | <i>F1</i> -score | Time (s) |
|------------|----------|--------|------------------|----------|
| 15 minutes | 0.861    | 0.865  | 0.863            | 0.0312   |
| 30 minutes | 0.878    | 0.882  | 0.88             | 0.0312   |
| 1 hour     | 0.864    | 0.873  | 0.868            | 0.0312   |
| 2 hours    | 0.852    | 0.865  | 0.858            | 0.0574   |
| 12 hours   | 0.729    | 0.706  | 0.717            | 0.0621   |

We use Algorithm 2 to build the information dissemination prediction model. In this algorithm, we assume that each user forwards the Weibo once and the publisher will not forward the Weibo.

## 4. Experiments and Results

In this section, the predicting performance is evaluated by using ELM. In addition, we compared the results between ELM and SVM based on adding the new feature we proposed and do not use the new feature. We also test the proposed information propagation prediction model and give it performance in this section.

**4.1. Users Behavior Prediction.** According to the data we crawl from Sina Weibo, we select 133190 forward data as the forward sample. According to Section 3.1, the numbers of each ignore sample are shown in Table 2.

We use ELM to forecast forward or ignore behavior of users. The source code of ELM can be obtained from the website (ELM Source Codes: ELM Source Codes: <http://www.ntu.edu.sg/home/egbhuang/>). We also compare the results between ELM and SVM. The tool of *lib-SVM* is used in this paper, which can be obtained from the website (data set: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>). In order to evaluate the effect of forecast model, we choose the evaluation index of information retrieval, including accuracy, recall, and the value of *F1*. With 10 times of cross validation method validation algorithm, we get the user forward behavior prediction results shown in Tables 3 and 4. Table 3 shows the performance using ELM and Table 4 shows the performance using SVM.

```

Input: Weibo publisher  $U_0$ .
Output: Forwarded number  $C$ .
(1)  $C = 0$ ;
(2) Read the fans list  $L(N)$  of  $U_0$  ( $N$  is the number of fans,  $N \geq 0$ )
(3) for ( $i = 0; i < N; i++$ )
(4)     if(the user  $L(i)$  is predicted as forwarding user)
(5)          $C = C + 1$ ;
(6)         Information_forecast( $L(i)$ );

```

ALGORITHM 2: Information\_forecast( $U$ ).

TABLE 4: User forward behavior prediction results using SVM.

| $\Delta t$ | Accuracy | Recall | $FI$ -score | Time (s) |
|------------|----------|--------|-------------|----------|
| 15 minutes | 0.867    | 0.875  | 0.871       | 0.0983   |
| 30 minutes | 0.869    | 0.88   | 0.874       | 0.0983   |
| 1 hour     | 0.855    | 0.862  | 0.858       | 0.0983   |
| 2 hours    | 0.846    | 0.861  | 0.853       | 0.1492   |
| 12 hours   | 0.749    | 0.743  | 0.745       | 0.2094   |

TABLE 5: Predicted results without the new feature using ELM.

| $\Delta t$ | Accuracy | Recall | $FI$ -score |
|------------|----------|--------|-------------|
| 15 minutes | 0.854    | 0.86   | 0.857       |
| 30 minutes | 0.858    | 0.869  | 0.863       |
| 1 hour     | 0.847    | 0.866  | 0.856       |
| 2 hours    | 0.836    | 0.858  | 0.847       |
| 12 hours   | 0.702    | 0.736  | 0.719       |

If we compare Tables 3 and 4, we can find ELM has a better performance than SVM. No matter what algorithm we used, when we take  $\Delta t$  as 30 minutes, we get the best performance. Because 15 minutes is too short, some people may not have had time to release or forward Weibo. People will not spend much time in browsing Weibo once. So when the  $\Delta t$  is taken as 2 hours, the performance is much lower than 30 minutes. We can also see that the performance of 12 hours is the lowest. This means people ignore Weibos not only because they do not like it, but also because they are not online. When  $\Delta t$  is too long, the absent behavior plays a dominant role.

At the same time, in order to consider the time factor, we also measured the running time of ELM and SVM. And the time of ELM is far lower than the SVM.

In order to test the effectiveness of the feature we proposed, we also test the performance without the new feature. Tables 5 and 6 show the predicted results without the feature we proposed.

We can see Tables 5 and 6 also have the same conclusion with Tables 3 and 4, in which 30 minutes has the highest performance. So when we do the information propagation prediction, we choose the dataset whose  $\Delta t$  is taken as 30 minutes. By comparing the Tables 3 and 5, we find using the new feature we proposed has a better performance than without the new feature. This can also be found by comparing Tables 4 and 6. In order to give a more intuitive description of

TABLE 6: Predicted results without the new feature using SVM.

| $\Delta t$ | Accuracy | Recall | $FI$ -score |
|------------|----------|--------|-------------|
| 15 minutes | 0.849    | 0.852  | 0.850       |
| 30 minutes | 0.856    | 0.859  | 0.857       |
| 1 hour     | 0.842    | 0.854  | 0.848       |
| 2 hours    | 0.828    | 0.84   | 0.834       |
| 12 hours   | 0.692    | 0.726  | 0.709       |

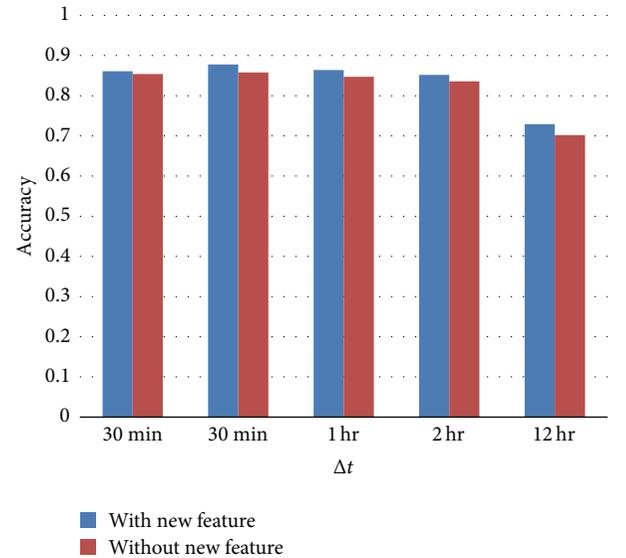


FIGURE 7: Comparison charts of using ELM.

this conclusion, we draw the figures to show the details. And Figures 7 and 8 show comparison charts of using ELM and using SVM.

As can be seen from the Figures 7 and 8, when using the dependency between the Weibos involved geographical locations and location of the user feature, the prediction results are better than without the feature.

To give a more intuitive description of the comparison, we also show the performance of ELM and SVM in a figure. Because 30 minutes has the best performance in both algorithm, we only compare the performance in this case. Figure 9 shows the comparison between ELM and SVM.

We can see in both cases that the predicted results obtained by ELM are higher than the SVM prediction results.

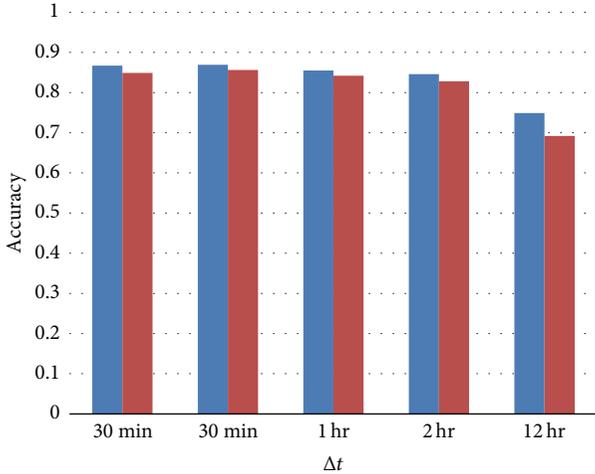


FIGURE 8: Comparison charts of using SVM.

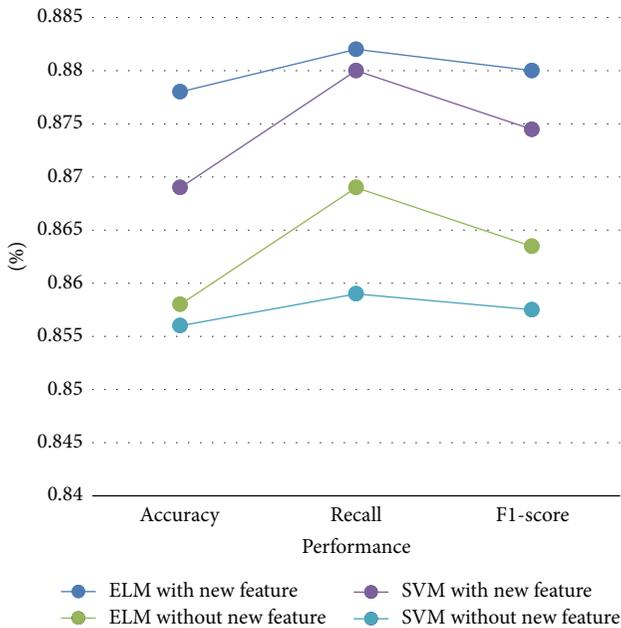


FIGURE 9: Comparison between ELM and SVM.

This proves that using ELM algorithm is better than using SVM algorithm. ELM algorithm has good performance. We can also see the new feature brings better performance.

4.2. Information Propagation Prediction. According to the algorithm in Section 3.3 of and prediction results of ELM, we predict the scale of the Information propagation. We choose 30000 original Weibos of 15375 users to verify our model. We count average user forward quantity proportion in every jump from the initial release users (jump: the shortest distance from users to the initial release user). Figure 10 shows the average of users' forward percentage in each jump.

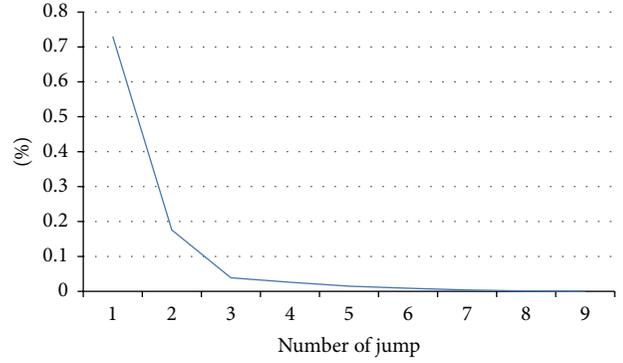


FIGURE 10: The average users' forward quantity proportion of each jump.

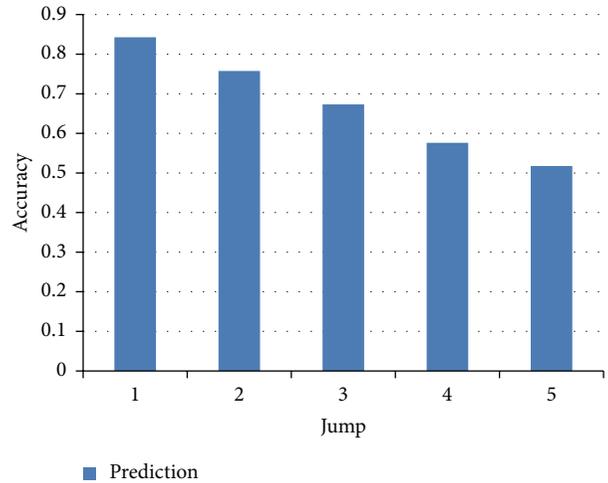


FIGURE 11: The accuracy in every jump.

It can be seen from Figure 10 that after 5 jump, the percent approach to 0. This proves that in our dataset all the forward behaviors happen at the first five jumps. This proves that the Weibo is a widely spread but deep low social network.

Based on this theory, our information propagation prediction stops at the fifth jump. This can avoid the excessive iteration. Figure 11 shows the accuracy we predict in every jump. The accuracy of jump 1 is the accuracy of the first forward layer of 30000 Weibos. Others are the same.

We can see in Figure 11 that the accuracy of the first jump is the highest. Accuracy reduces with the increase of the jump count. This is because when we do the prediction, the error is constantly accumulated. When the jump comes to 5, the error has been accumulated to a considerable scale. So the accuracy becomes very low.

In order to determine the scale of information dissemination, we divide the scale according to the  $10^n$  order of magnitudes. If the information dissemination scale we predicted is in the same order of magnitude which is the actual information dissemination scale, we can say the prediction is right. We calculated the average predict information dissemination

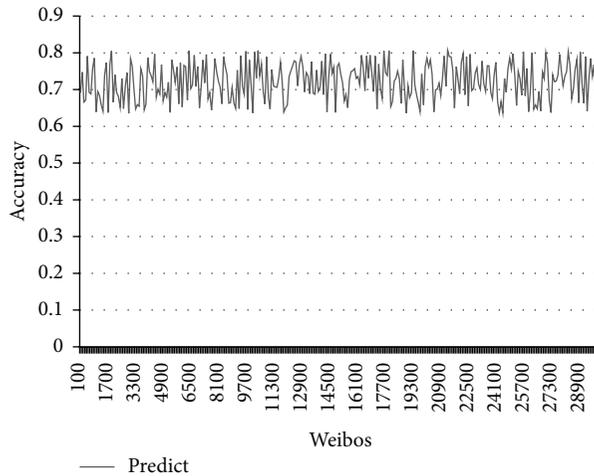


FIGURE 12: Average prediction accuracy of each user.

scale accuracy of 30000 Weibos. Figure 12 shows the average prediction accuracy of each Weibo.

As can be seen from Figure 12, for different Weibo from different users, our algorithm accuracy is around 70%. This is because for the Weibos whose forward deep close to 5 or more than 5 jumps, the error of our model has been accumulated to a considerable scale and brings decline in accuracy.

For the selected data, our predicting result is very stable. This proves that our algorithm is real and effective.

## 5. Conclusions

Online behavior of Weibo users and information dissemination analysis is a hot issue nowadays. In this paper, we analyzed the features of Sina Weibo user behavior and predicted the information transmission. We proposed 8 features to analyze user behavior. They are the dependency between the Weibos involved geographical locations and location of the user, the influence of user, user release activity, user forward activity, the intimacy between the users, the interest similarity between user and content, the interest similarity between users, and Weibo content importance. The feature (i.e., the dependency between the Weibos involved geographical locations and locations of the users) is the new feature we proposed. We used ELM to analyze if users will forward or ignore a weibo. Our experiment results show that the feature we proposed is very effective and ELM gets better results than SVM. We also test the different performance between the different values of  $\Delta t$  in ignore dataset. We found that when  $\Delta t$  is 30 minutes, the performance is the best. So we use the 30 minutes ignore dataset to build the training set. Based on that, we proposed information propagation prediction model and calculate the scale of the information propagation. The experiment results show that our model has a good performance.

The features and model we proposed in this paper can give some help to businesses and government. They can use our model to predict the scale of the information propagation before they publish it. If the scale is small, they can use our

feature to adjust the information text. The model and features has very high practical value.

However, there is still something we need to improve in this paper. For example, when considering information dissemination size, we do not concern users forward their own Weibo and people may forward the Weibo many times. We will take it into consideration in the future.

## Conflict of Interests

The researchers claim no conflict of interests.

## Acknowledgments

This research was partially supported by the National Natural Science Foundation of China under Grant nos. 61332006 and 61100022, the National Basic Research Program of China under Grant no. 2011CB302200-G, and the 863 Program under Grant no. 2012AA011004.

## References

- [1] G. Ou, W. Chen, B. Li, T. Wang, D. Yang, and K.-F. Wong, "CLUSM: an unsupervised model for microblog sentiment analysis incorporating link information," in *Database Systems for Advanced Applications*, vol. 8421 of *Lecture Notes in Computer Science*, pp. 481–494, Springer, 2014.
- [2] J. Sun and Y. Zhu, "Microblogging personalized recommendation based on ego networks," in *Proceedings of the 12th IEEE/WIC/ACM International Conference on Web Intelligence (WI) and Intelligent Agent Technologies (IAT '13)*, pp. 165–170, Atlanta, Ga, USA, November 2013.
- [3] G. Song, Z. Li, and H. Tu, "Forward or ignore: user behavior analysis and prediction on microblogging," in *Advanced Research in Applied Artificial Intelligence*, vol. 7345 of *Lecture Notes in Computer Science*, pp. 231–241, Springer, Berlin, Germany, 2012.
- [4] J.-X. Cao, J.-L. Wu, W. Shi, B. Liu, X. Zheng, and J.-Z. Luo, "Sina microblog information diffusion analysis and prediction," *Chinese Journal of Computers*, vol. 37, no. 4, pp. 779–790, 2014.
- [5] A. Mogadala and V. Varma, "Twitter user behavior understanding with mood transition prediction," in *Proceedings of the ACM Workshop on Data-Driven User Behavioral Modeling and Mining from Social Media (DUBMMSM '12)*, pp. 31–34, October 2012.
- [6] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Patterns of cascading behavior in large blog graphs," in *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 551–556, Philadelphia, PA, USA, April 2007.
- [7] J. Lingad, S. Karimi, and J. Yin, "Location extraction from disaster-related microblogs," in *Proceedings of the 22nd International Conference on World Wide Web Companion (WWW '13)*, pp. 1017–1020, 2013.
- [8] S. Hosseini, S. Unankard, X. Zhou, and S. Sadiq, "Location oriented phrase detection in microblogs," in *Database Systems for Advanced Applications: Proceedings of the 19th International Conference, DASFAA 2014, Bali, Indonesia, April 21–24, 2014, Part I*, vol. 8421 of *Lecture Notes in Computer Science*, pp. 495–509, Springer International Publishing, Cham, Switzerland, 2014.

- [9] J. Marques and C. Serrão, "Improving user content privacy on social networks using rights management systems," *Annals of Telecommunications*, vol. 69, no. 1-2, pp. 37–45, 2014.
- [10] H. T. Quang, H. V. H. Tien, H. N. Le, T. H. Trung, and P. Do, "Finding the cluster of actors in social network based on the topic of messages," in *Intelligent Information and Database Systems*, vol. 8397 of *Lecture Notes in Computer Science*, pp. 183–190, Springer, New York, NY, USA, 2014.
- [11] C.-Y. Tseng and M.-S. Chen, "Incremental SVM model for spam detection on dynamic email social networks," in *Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE '09)*, pp. 128–135, Vancouver, Canada, August 2009.
- [12] T. R. Zaman, R. Herbrich, J. van Gael, and D. Stern, "Predicting information spreading in twitter," in *Proceedings of the Workshop on Computational Social Science and the Wisdom of Crowds*, NIPS 2010, 2010.
- [13] F. Wu and B. A. Huberman, "Novelty and collective attention," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 45, pp. 17599–17601, 2007.
- [14] W. Webberley, S. Allen, and R. Whitaker, "Retweeting: a study of message-forwarding in Twitter," in *Proceedings of the 1st IEEE NSS Workshop on Mobile and Online Social Networks (MOSN '11)*, pp. 13–18, Milan, Italy, September 2011.
- [15] R. Bandari, S. Asur, and B. A. Huberman, "The pulse of news in social media: forecasting popularity," in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM '12)*, pp. 26–33, Dublin, Ireland, June 2012.
- [16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '04)*, pp. 985–990, Budapest, Hungary, July 2004.
- [17] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [18] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [19] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [20] D. Ren, X. Zhang, Z. Wang, J. Li, and X. Yuan, "Weibo events: a crowd sourcing weibo visual analytic system," in *Proceedings of the 7th IEEE Pacific Visualization Symposium (PacificVis '14)*, pp. 330–334, Yokohama, Japan, March 2014.
- [21] S. Petrović, M. Osborne, and V. Lavrenko, "RT to win! predicting message propagation in twitter," in *Proceedings of the 5th International Conference on Weblogs and Social Media (ICWSM '11)*, pp. 586–589, Barcelona, Spain, July 2011.
- [22] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: bringing order to the web," Tech. Rep. SIDL-WP, Stanford University, 1999.
- [23] X.-M. Lin and W. Wang, "Set and string similarity queries: a survey," *Chinese Journal of Computer*, vol. 34, no. 10, pp. 1853–1862, 2011.
- [24] C. Shi, C. Xu, and X. Yang, "Study of TFIDF algorithm," *Journal of Computer Applications*, vol. 6, no. 29, pp. 167–170, 2009.

## Research Article

# Particle Swarm Optimization Based Selective Ensemble of Online Sequential Extreme Learning Machine

Yang Liu,<sup>1</sup> Bo He,<sup>1</sup> Diya Dong,<sup>1</sup> Yue Shen,<sup>1</sup> Tianhong Yan,<sup>2</sup>  
Rui Nian,<sup>1</sup> and Amaury Lendasse<sup>3,4</sup>

<sup>1</sup>School of Information Science and Engineering, Ocean University of China, 238 Songling Road, Qingdao 266100, China

<sup>2</sup>School of Mechanical and Electrical Engineering, China Jiliang University, 258 Xueyuan Street, Xiasha High-Edu Park, Hangzhou 310018, China

<sup>3</sup>Department of Mechanical and Industrial Engineering and the Iowa Informatics Initiative, 3131 Seamans Center, The University of Iowa, Iowa City, IA 52242-1527, USA

<sup>4</sup>Arcada University of Applied Sciences, 00550 Helsinki, Finland

Correspondence should be addressed to Bo He; [bhe@ouc.edu.cn](mailto:bhe@ouc.edu.cn) and Tianhong Yan; [thyan@163.com](mailto:thyan@163.com)

Received 7 August 2014; Revised 1 November 2014; Accepted 5 November 2014

Academic Editor: Zhan-li Sun

Copyright © 2015 Yang Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A novel particle swarm optimization based selective ensemble (PSOSEN) of online sequential extreme learning machine (OS-ELM) is proposed. It is based on the original OS-ELM with an adaptive selective ensemble framework. Two novel insights are proposed in this paper. First, a novel selective ensemble algorithm referred to as particle swarm optimization selective ensemble is proposed, noting that PSOSEN is a general selective ensemble method which is applicable to any learning algorithms, including batch learning and online learning. Second, an adaptive selective ensemble framework for online learning is designed to balance the accuracy and speed of the algorithm. Experiments for both regression and classification problems with UCI data sets are carried out. Comparisons between OS-ELM, simple ensemble OS-ELM (EOS-ELM), genetic algorithm based selective ensemble (GASEN) of OS-ELM, and the proposed particle swarm optimization based selective ensemble of OS-ELM empirically show that the proposed algorithm achieves good generalization performance and fast learning speed.

## 1. Introduction

Feedforward neural network is one of the most prevailing neural networks for data processing in the past decades [1, 2]. However, the slow learning speed limits its applications. Recently, an original algorithm designed for single hidden layer feedforward neural networks (SLFNs) named extreme learning machine (ELM) was proposed by Huang et al. [3]. ELM is a tuning free algorithm for it randomly selects the input weights and biases of the hidden nodes instead of learning these parameters. And, also, the output weights of the network are then analytically determined. ELM proves to be a few orders faster than traditional learning algorithms and obtains better generalization performance as well. It lets the fast and accurate data analytics become possible and has been applied to many fields [4–6].

However, the algorithms mentioned above need all the training data available to build the model, which is referred to as batch learning. In many industrial applications, it is very common that the training data can only be obtained one by one or chunk by chunk. If batch learning algorithms are performed each time new training data is available, the learning process will be very time consuming. Hence online learning is necessary for many real world applications.

An online sequential extreme learning machine is then proposed by Liang et al. [7]. OS-ELM can learn the sequential training observations online at arbitrary length (one by one or chunk by chunk). New arrived training observations are learned to update the model of the SLFNs. As soon as the learning procedure for the arrived observations is completed, the data is discarded. Moreover, it has no prior knowledge about the amount of the observations which

will be presented. Therefore, OS-ELM is an elegant online learning algorithm which can handle both the RBF and additive nodes in the same framework and can be used to both the classification and function regression problems. OS-ELM proves to be a very fast and accurate online sequential learning algorithm [8–10], which can provide better generalization performance in faster speed compared with other online learning algorithms such as GAP-RBF, GGAP-RBF, SGBP, RAN, RANEKE, and MRAN.

However, due to the random generation of the parameters for the hidden nodes, the generalization performance of OS-ELM sometimes cannot be guaranteed, similar to ELM. Some ensemble based methods have been applied to ELM to improve its accuracy [11–13]. Ensemble learning is a learning scheme where a collection of a finite number of learners are trained for the same task [14, 15]. It has been demonstrated that the generalization ability of a learner can be significantly improved by ensembling a set of learners. In [16] a simple ensemble OS-ELM, that is, EOS-ELM, has been investigated. However, Zhou et al. [17] proved that selective ensemble is a better choice. We apply this idea to OS-ELM. At first, a novel selective ensemble algorithm, termed as PSosen, is proposed. PSosen adopts particle swarm optimization [18] to select the individual OS-ELMs to form the ensemble. Benefiting from the fast speed of PSO, PSosen is designed to be a new accurate and fast selective ensemble algorithm. It should be noted that PSosen is a general selective ensemble algorithm suitable for any learning algorithms.

Different from batch learning, online learning algorithms need to perform learning continually. Therefore the complexity of the learning algorithm should be taken into account. Obviously, performing selective ensemble learning each step is not a good choice for online learning. Thus we designed an adaptive selective ensemble framework for OS-ELM. A set of OS-ELMs are trained online, and the root mean square error (RMSE) will always be calculated. The error will be compared with a preset threshold  $\lambda$ . If RMSE is bigger than the threshold, it means the model is not accurate. Then PSosen will be performed and a selective ensemble  $M$  is obtained. Otherwise, it means the model is relatively accurate and the ensemble will not be selected. Then the output of the system is calculated as the average of the individuals in the ensemble set. And each individual OS-ELM will be updated recursively.

UCI data sets [19], which contain both regression and classification data, are used to verify the feasibility of the proposed algorithm. Comparisons of three aspects including RMSE, standard deviation and running time between OS-ELM, and EOS-ELM, selective ensemble of OS-ELM (SEOS-ELM) with both GASEN and PSosen are presented. The results convincingly show that PSosen achieves better generalization accuracy and fast learning speed.

The rest of the paper is organized as follows. In Section 2, previous work including ELM and OS-ELM is reviewed. The novel selective ensemble based on particle swarm optimization is presented in Section 3. An adaptive selective ensemble framework is designed for OS-ELM in Section 4. Experiments are carried out in Section 5 and the comparison results are also presented. In Section 6, further discussion

about PSosen is provided. We draw the conclusion of the paper in Section 7.

## 2. Review of Related Work

In this section, both the basic ELM algorithm and the online version OS-ELM are reviewed in brief as the background knowledge for our work.

*2.1. Extreme Learning Machine (ELM).* ELM algorithm is derived from single hidden layer feedforward neural networks (SLFNs). Unlike traditional SLFNs, ELM assigns the parameters of the hidden nodes randomly without any iterative tuning. Besides, all the parameters of the hidden nodes in ELM are independent of each other. Hence ELM can be seen as generalized SLFNs.

Given  $N$  training samples  $(x_i, t_i) \in R^n \times R^m$ , where  $x_i$  is an input vector of  $n$  dimensions and  $t_i$  is a target vector of  $m$  dimensions. Then SLFNs with  $\bar{N}$  hidden nodes each with output function  $G(a_i, b_i, x)$  are mathematically modeled as

$$f_{\bar{N}}(x_j) = \sum_{i=1}^{\bar{N}} \beta_i G(a_i, b_i, x_j) = t_j, \quad j = 1, \dots, N, \quad (1)$$

where  $(a_i, b_i)$  are parameters of hidden nodes, and  $\beta_i$  is the weight vector connecting the  $i$ th hidden node and the output node. To simplify, (1) can be written equivalently as

$$H\beta = T, \quad (2)$$

where

$$H(a_1, \dots, a_N, b_1, \dots, b_{\bar{N}}, x_1, \dots, x_N) = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_{\bar{N}}, b_{\bar{N}}, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_{\bar{N}}, b_{\bar{N}}, x_N) \end{bmatrix}_{N \times \bar{N}}, \quad (3)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\bar{N}}^T \end{bmatrix}_{\bar{N} \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m},$$

$H$  is called the hidden layer output matrix of the neural network, and the  $i$ th column of  $H$  is the output of the  $i$ th hidden node with respect to inputs  $x_1, x_2, \dots, x_N$ .

In ELM,  $H$  can be easily obtained as long as the training set is available and the parameters  $(a_i, b_i)$  are randomly assigned. Then ELM evolves into a linear system and the output weights  $\beta$  are calculated as

$$\hat{\beta} = H^\dagger T, \quad (4)$$

where  $H^\dagger$  is the Moore-Penrose generalized inverse of matrix  $H$ .

The ELM algorithm can be summarized in three steps as shown in Algorithm 1.

Input:

A training set  $N = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \dots, N\}$ , hidden node output function  $G(a_i, b_i, x)$ , and the number of hidden nodes  $\tilde{N}$ .

Steps:

(1) Assign parameters of hidden nodes  $(a_i, b_i)$  randomly,  $i = 1, \dots, \tilde{N}$ .

(2) Calculate the hidden layer output matrix  $H$ .

(3) Calculate the output weight  $\beta : \hat{\beta} = H^\dagger T$ , where  $H^\dagger$  is the Moore-Penrose generalized inverse of hidden layer output matrix  $H$ .

#### ALGORITHM 1

2.2. OS-ELM. In many industrial applications, it is impossible to have all the training data available before the learning process. It is common that the training observations are sequentially inputted to the learning algorithm; that is, the observations arrive one-by-one or chunk-by-chunk. In this case, the batch ELM algorithm is no longer applicable. Hence, a fast and accurate online sequential extreme learning machine was proposed to deal with online learning.

The output weight  $\beta$  obtained from (4) is actually a least-squares solution of (2). Given  $\text{rank}(H) = \tilde{N}$ , the number of hidden nodes,  $H^\dagger$  can be presented as

$$H^\dagger = (H^T H)^{-1} H^T. \quad (5)$$

This can also be called the left pseudoinverse of  $H$  for it satisfies the equation  $H^\dagger H = I_{\tilde{N}}$ . If  $H^T H$  tends to be singular, smaller network size  $\tilde{N}$  and larger data number  $N_0$  should be chosen in the initialization step of OS-ELM. Substituting (5) to (4), we can get

$$\hat{\beta} = (H^T H)^{-1} H^T T \quad (6)$$

which is the least-squares solution to (2). Then the OS-ELM algorithm can be deduced by recursive implementation of the least-squares solution of (6).

There are two main steps in OS-ELM, initialization step and update step. In the initialization step, the number of training data  $N_0$  needed in this step should be equal to or larger than network size  $\tilde{N}$ . In the update step, the learning model is updated with the method of recursive least square (RLS). And only the newly arrived single or chunk training observations are learned, which will be discarded as soon as the learning step is completed.

The two steps for OS-ELM algorithm in general are as follows.

- (a) Initialization step: batch ELM is used to initialize the learning system with a small chunk of initial training data  $\mathbb{N}_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$  from given training set

$$\mathbb{N} = \{(x_i, t_i), x_i \in R^n, t_i \in R^m, i = 1, \dots\}, \quad N_0 \geq \tilde{N}. \quad (7)$$

- (1) Assign random input weights  $a_i$  and bias  $b_i$  (for additive hidden nodes) or center  $a_i$  and impact factor  $b_i$  (for RBF hidden nodes),  $i = 1, \dots, \tilde{N}$ .

- (2) Calculate the initial hidden layer output matrix:

$$H_0 = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_{N_0}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_{N_0}) \end{bmatrix}_{N_0 \times \tilde{N}}. \quad (8)$$

- (3) Calculate the initial output weight  $\beta^{(0)} = P_0 H_0^T T_0$ , where  $P_0 = (H_0^T H_0)^{-1}$  and  $T_0 = [t_1, \dots, t_{N_0}]^T$ .

- (4) Set  $k = 0$ . Initialization is finished.

- (b) Sequential learning step is as follows.

The  $(k + 1)$ th chunk of new observations can be expressed as

$$\mathbb{N}_{k+1} = \{(x_i, t_i)\}_{i=(\sum_{j=0}^k N_j)+1}^{\sum_{j=0}^{k+1} N_j}, \quad (9)$$

where  $\mathbb{N}_{k+1}$  represents the number of observations in the  $(k + 1)$ th chunk newly arrived.

- (1) Compute the partial hidden layer output matrix  $H_{k+1}$  for the  $(k + 1)$ th chunk:

$$H_{k+1} = \begin{bmatrix} G(a_1, b_1, x_{(\sum_{j=0}^k N_j)+1}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_{(\sum_{j=0}^k N_j)+1}) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_{\sum_{j=0}^{k+1} N_j}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_{\sum_{j=0}^{k+1} N_j}) \end{bmatrix}_{N_{k+1} \times \tilde{N}}. \quad (10)$$

- (2) Set  $T_{k+1} = [t_{(\sum_{j=0}^k N_j)+1}, \dots, t_{\sum_{j=0}^{k+1} N_j}]^T$ . And we have

$$K_{k+1} = K_k + H_{k+1}^T H_{k+1}, \quad (11)$$

$$\beta^{(k+1)} = \beta^{(k)} + K_{k+1}^{-1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^{(k)}).$$

To avoid calculating inverse in the iterative procedure,  $K_{k+1}^{-1}$  is factored as the following according to Woodbury formula:

$$\begin{aligned} K_{k+1}^{-1} &= (K_k + H_{k+1}^T H_{k+1})^{-1} \\ &= K_k^{-1} - K_k^{-1} H_{k+1}^T (I + H_{k+1} K_k^{-1} H_{k+1}^T)^{-1} H_{k+1} K_k^{-1}. \end{aligned} \quad (12)$$

Let  $P_{k+1} = K_{k+1}^{-1}$ .

- (3) Calculate the output weight  $\beta^{(k+1)}$ , according to the updating equations:

$$\begin{aligned} P_{k+1} &= P_k - P_k H_{k+1}^T \left( I + H_{k+1} P_k H_{k+1}^T \right)^{-1} H_{k+1} P_k, \\ \beta^{(k+1)} &= \beta^{(k)} + P_{k+1} H_{k+1}^T \left( T_{k+1} - H_{k+1} \beta^{(k)} \right). \end{aligned} \quad (13)$$

- (4) Set  $k = k + 1$ . Go to step (b).

### 3. Particle Swarm Optimization Selective Ensemble

In this section, a novel selective ensemble method referred to as particle swarm optimization selective ensemble (PSOSEN) is proposed. PSOSEN adopts particle swarm optimization to select the good learners and combine their predictions. Detailed procedures of the PSOSEN algorithm will be introduced in this section.

A remarkable superiority of PSOSEN is its speed over other selective ensemble algorithms. Another popular selective ensemble learning method is based on genetic algorithm. Compared with GASEN, PSOSEN achieves faster convergence to optimal solution due to the omission of crossover and mutation operations used in GASEN. GASEN is actually quite complicated for the requirement of encode, decode, and other genetic operations. For instance, GASEN only works with binary encoding, while PSOSEN is available for any forms of values based on their current positions and velocity vectors in the corresponding hyperspace. For PSOSEN, there is no need for overmuch parameter adjustment, thus easy to implement. Although using simple method, PSOSEN is still capable of obtaining high accuracy of prediction and reaching the optima earlier than GASEN. Furthermore, PSO is less influenced by changes in problem dimensionality or modality of problems compared with GA, which also proves to be robust in most situations [20].

As selective ensemble is usually more time-consuming than original algorithm, a faster optimization method might be preferable. For this purpose, PSOSEN might be more appropriate to be adopted to search for the optimal ensemble of ELM models efficiently.

Zhou et al. [17] have demonstrated that ensembling many of the available learners may be better than ensembling all of those learners in both regression and classification. The detailed proof of this conclusion will not be presented in this paper. However, one important problem for selective ensemble is how to select the good learners in a set of available learners.

The novel approach selective ensemble algorithm is proposed to select good learners in the ensemble. PSOSEN is based on the idea of heuristics. It assumes each learner can be assigned a weight, which could characterize the fitness of including this learner in the ensemble. Then the learner with the weight bigger than a preset threshold  $\lambda$  could be selected to join the ensemble.

We will explain the principle of PSOSEN from the context of regression. We use  $\omega_i$  to denote the weight of the  $i$ th

component learner. The weight should satisfy the following equations:

$$\begin{aligned} 0 &\leq \omega_i \leq 1, \\ \sum_{i=1}^N \omega_i &= 1. \end{aligned} \quad (14)$$

Then the weight vector is

$$\omega = (\omega_1, \omega_2, \dots, \omega_N). \quad (15)$$

Suppose input variables  $x \in R^m$  according to the distribution  $p(x)$ , the true output of  $x$  is  $d(x)$ , and the actual output of the  $i$ th learner is  $f_i(x)$ . Then the output of the simple weighted ensemble on  $x$  is

$$\hat{f}(x) = \sum_{i=1}^N \omega_i f_i(x). \quad (16)$$

Then the generalization error  $E_i(x)$  of the  $i$ th learner and the generalization error  $\hat{E}(x)$  of the ensemble are calculated on  $x$ , respectively:

$$\begin{aligned} E_i(x) &= (f_i(x) - d(x))^2, \\ \hat{E}(x) &= (\hat{f}(x) - d(x))^2. \end{aligned} \quad (17)$$

The generalization error  $E_i$  of the  $i$ th learner and that of the ensemble  $\hat{E}$  is calculated on  $p(x)$ , respectively:

$$\begin{aligned} E_i &= \int dx p(x) E_i(x), \\ \hat{E} &= \int dx p(x) \hat{E}(x). \end{aligned} \quad (18)$$

We then define the correlation between the  $i$ th and the  $j$ th component learner as follows:

$$C_{ij} = \int dx p(x) (f_i(x) - d(x)) (f_j(x) - d(x)). \quad (19)$$

Obviously  $C_{ij}$  satisfies the following equations:

$$\begin{aligned} C_{ii} &= E_i, \\ C_{ij} &= C_{ji}. \end{aligned} \quad (20)$$

Considering the equations defined above, we can get

$$\hat{E}(x) = \left( \sum_{i=1}^N \omega_i f_i(x) - d(x) \right) \left( \sum_{j=1}^N \omega_j f_j(x) - d(x) \right), \quad (21)$$

$$\hat{E} = \sum_{i=1}^N \sum_{j=1}^N \omega_i \omega_j C_{ij}. \quad (22)$$

To minimize the generalization error of the ensemble, according to (22), the optimum weight vector can be obtained as

$$\omega_{\text{opt}} = \underset{\omega}{\operatorname{argmin}} \left( \sum_{i=1}^N \sum_{j=1}^N \omega_i \omega_j C_{ij} \right). \quad (23)$$

The  $k$ th variable of  $\omega_{\text{opt}}$ , that is,  $\omega_{\text{opt},k}$ , can be solved by Lagrange multiplier:

$$\frac{\partial \left( \sum_{i=1}^N \sum_{j=1}^N \omega_i \omega_j C_{ij} - 2 * \lambda \left( \sum_{i=1}^N \omega_i - 1 \right) \right)}{\partial \omega_{\text{opt},k}} = 0. \quad (24)$$

The equation can be simplified to

$$\sum_{\substack{j=1 \\ j \neq k}}^N \omega_{\text{opt},k} C_{kj} = \lambda. \quad (25)$$

Taking (2) into account, we can get

$$\omega_{\text{opt},k} = \frac{\sum_{j=1}^N C_{kj}^{-1}}{\sum_{i=1}^N \sum_{j=1}^N \omega_i \omega_j C_{ij}^{-1}}. \quad (26)$$

Equation (26) gives the direct solution for  $\omega_{\text{opt}}$ . But the solution seldom works well in real world applications. Due to the fact that some learners are quite similar in performance, when a number of learners are available, the correlation matrix  $C_{ij}$  may be irreversible or ill-conditioned.

Although we cannot obtain the optimum weights of the learner directly, we can approximate them in some way. Equation (23) can be viewed as an optimization problem. As particle swarm optimization has been proved to be a powerful optimization tool, PSOSEN is then proposed. The basic PSO algorithm is shown in Figure 1.

PSOSEN randomly assigns a weight to each of the available learners at first. Then it employs particle swarm optimization algorithm to evolve those weights so that the weights can characterize the fitness of the learners in joining the ensemble. Finally, learners whose weight is bigger than a preset threshold  $\lambda$  are selected to form the ensemble. Note that if all the evolved weights are bigger than the threshold  $\lambda$ , then all the learners will be selected to join the ensemble.

PSOSEN can be applied to both regression and classification problems for the purpose of the weights evolving process which is only to select the component learners. In particular, the outputs of the ensemble for regression are combined via simple averaging instead of weighted averaging. The reason is that previous work [17] showed that using the weights both in selection of the component learners and in combination of the outputs tends to suffer the overfitting problem.

In the process of generating population, the goodness of the individuals is evaluated via validation data bootstrap sampled from the training data set. We use  $\hat{E}_\omega^V$  to denote the generalization error of the ensemble, which corresponds to individual  $\omega$  on the validation data  $V$ . Obviously  $\hat{E}_\omega^V$  can describe the goodness of  $\omega$ . The smaller  $\hat{E}_\omega^V$  is, the better  $\omega$  is. So, PSOSEN adopts  $f(\omega) = 1/\hat{E}_\omega^V$  as the fitness function.

The PSOSEN algorithm is summarized as follows.  $S_1, S_2, \dots, S_T$  are bootstrap samples generated from original training data set. A component learner  $N_i$  is trained from each  $S_T$ . And a selective ensemble  $N^*$  is built from  $N_1, N_2, \dots, N_T$ . The output is the average output of the ensemble for regression or the class label who receives the most number in voting process for classification (see Algorithm 2).

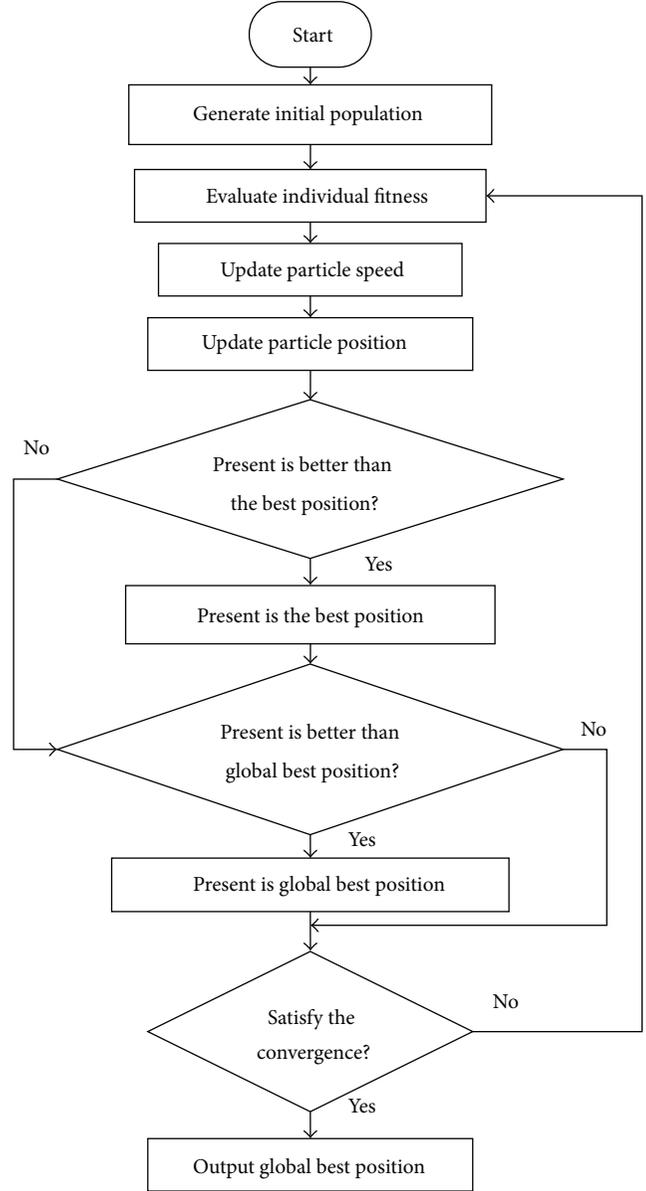
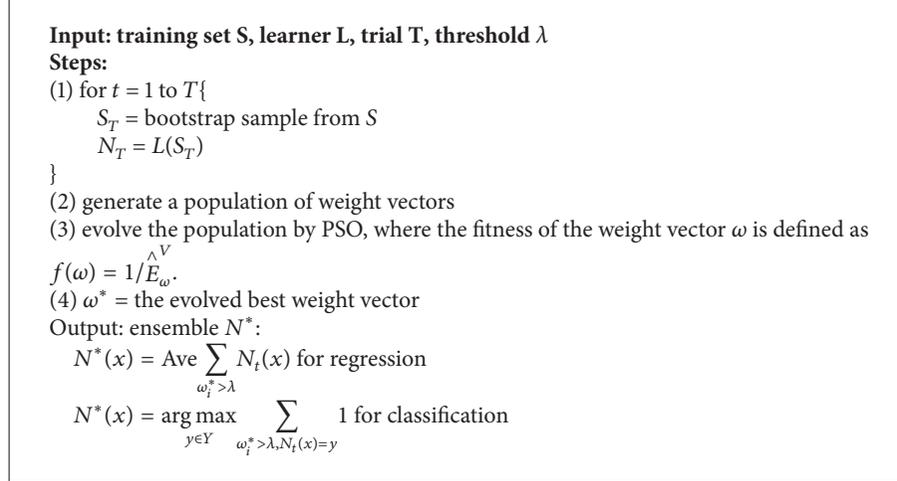


FIGURE 1: Flowchart for particle swarm optimization algorithm.

#### 4. Particle Swarm Optimization Based Selective Ensemble of Online Sequential Extreme Learning Machine

In this section, PSOSEN is applied to the original OS-ELM to improve the generalization performance. In order to reduce the complexity and employ PSOSEN flexibly, an adaptive framework is then designed. The flowchart of the framework is shown as in Figure 2.

Online learning is necessary in many industrial applications. In these situations, training data can only be obtained sequentially. Although OS-ELM is proposed as useful online learning algorithm, the generalization performance may not be quite good results from the random generation of the input



ALGORITHM 2: PBOSEN.

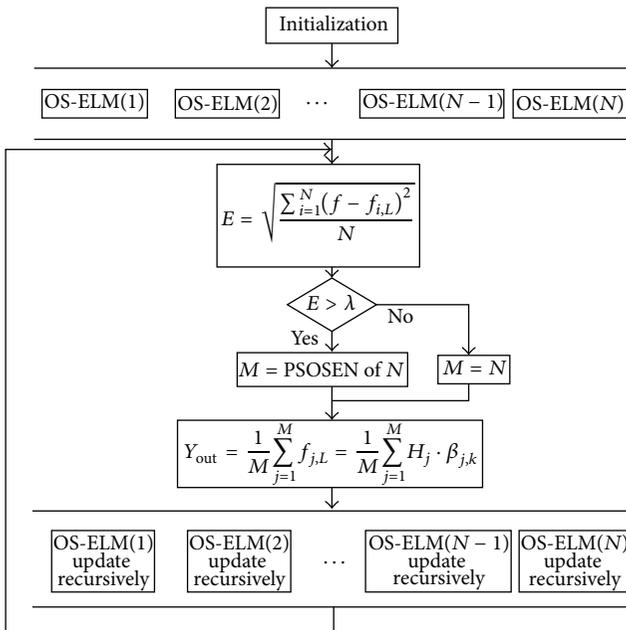


FIGURE 2: Flowchart for the proposed framework.

parameters. Ensemble methods have been investigated in OS-ELM, that is, the EOS-ELM algorithm [16]. However, it is only very simple ensemble method, which just calculates the average of all the  $N$  individual OS-ELMs. In this section, selective ensemble, which is superior to simple ensemble, is applied to OS-ELM. The novel selective ensemble method proposed in Section 3 is adopted. Apparently, performing PBOSEN each step is a time consuming process. We design an adaptive framework to determine whether to perform PBOSEN or simple ensemble. Thus the accuracy and the complexity can be balanced well. The framework for the new algorithm can be explained as follows.

First,  $N$  individual OS-ELMs are initialized. The number of nodes is same for each OS-ELM, while the input weights and biases for each OS-ELM are randomly generated.

Second, the RMSE error is calculated:

$$E = \sqrt{\frac{\sum_{i=1}^N (f - f_{i,L})^2}{N}}, \quad (27)$$

where  $f$  is the expected output, while  $f_{i,L}$  is the actual output of the  $i$ th individual OS-ELM.

The RMSE will be compared with a preset threshold  $\lambda$ . If  $E$  is bigger than  $\lambda$ , which means simple ensemble is not accurate, PBOSEN is performed and a selective ensemble  $M$  is obtained. And if  $E$  is smaller than  $\lambda$ , which indicates that simple ensemble is relatively accurate, the ensemble will not be selected.

Third, the output of the system is calculated as the average output of the individual in the ensemble set:

$$Y_{\text{out}} = \frac{1}{M} \sum_{j=1}^M f_{j,L} = \frac{1}{M} \sum_{j=1}^M H_j \cdot \beta_{j,k}, \quad (28)$$

where  $H_j$  is the output matrix of the  $j$ th OS-ELM, and  $\beta_{j,k}$  is the output weight calculated by the  $j$ th OS-ELM at step  $k$ .

At last, each OS-ELM will update recursively according to the update equations presented in Section 2.

## 5. Performance Evaluation of PBOSEN Based OS-ELM

In this section, a series of experiments were conducted to evaluate the performance of the proposed algorithm. OS-ELM, EOS-ELM, and GASEN based OS-ELM are also compared with the new algorithm in this section. All the experiments were carried out in the MATLAB R2012b environment on a desktop of CPU 3.40 GHz and 8 GB RAM.

**5.1. Model Selection.** For OS-ELM, the number of hidden nodes is the only parameter that needs to be determined. Cross-validation method is usually used to choose this parameter. Fifty trials of simulations are performed, respectively, for regression and classification problems. The number of hidden nodes is then determined by the validation error.

TABLE 1: Network selection for New-thyroid dataset.

| Number of networks | 1      | 5      | 10     | 15     | 20     | 25    | 30     |
|--------------------|--------|--------|--------|--------|--------|-------|--------|
| Testing accuracy   | 90.73  | 91.25  | 90.65  | 90.18  | 92.24  | 91.79 | 91.8   |
| Testing dev.       | 0.0745 | 0.0254 | 0.0316 | 0.0276 | 0.0138 | 0.024 | 0.0156 |

TABLE 2: Specification of benchmark datasets.

| Dataset                 | Classes | Training data | Testing data | Attributes |
|-------------------------|---------|---------------|--------------|------------|
| Regression problems     |         |               |              |            |
| Abalone                 | —       | 3000          | 1177         | 8          |
| California housing      | —       | 8000          | 12640        | 8          |
| Mackey-Glass            | —       | 4000          | 500          | 4          |
| Classification problems |         |               |              |            |
| New-thyroid             | 3       | 140           | 75           | 5          |
| Image segmentation      | 7       | 1500          | 810          | 19         |
| Satellite image         | 6       | 4435          | 2000         | 36         |

For EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN), there is another parameter that needs to be determined, that is, the number of networks in the ensemble. The parameter is set from 5 to 30 with the interval 5. Finally, the optimal parameter is selected according to the RMSE for regression, testing accuracy for classification, and standard deviation value. Under the same problem, the number of OS-ELMs is selected based on the lowest standard deviation and the comparable RMSE or accuracy compared with OS-ELM. Table 1 is an example of selecting the optimal number of networks for SEOS-ELM (PSOSEN) with RBF hidden nodes on New-thyroid dataset. As illustrated by Table 1, the lowest standard deviation occurs when the number of OS-ELMs is 20. Meanwhile, the prediction accuracy of SEOS-ELM is better than OS-ELM. Hence we set the number of networks to be 20 for the New-thyroid dataset. The numbers of OS-ELMs for other datasets are determined in the same way.

Both the Gaussian radial basis function (RBF)  $G(a, b, x) = \exp(-\|x - a\|^2/b)$  and the sigmoid additive  $G(a, b, x) = 1/(1 + \exp(-(a \cdot x + b)))$  are adopted as activation function in OS-ELM, EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN).

In the experiments, OS-ELM, EOS-ELM, and SEOS-ELM (GASEN) were compared with SEOS-ELM (PSOSEN). Some general information of the benchmark datasets used in our evaluations is listed in Table 2. Both regression and classification problems are included.

For OS-ELM, the input weights and biases with additive activation function or the centers with RBF activation function were all generated from the range  $[-1, 1]$ . For regression problems, all the inputs and outputs were normalized into the range  $[0, 1]$ , while the inputs and outputs were normalized into the range  $[-1, 1]$  for classification problems.

The benchmark datasets studied in the experiments are from UCI Machine Learning Repository except California

Housing dataset from the StatLib Repository. Besides, a time-series problem, Mackey-Glass, from UCI was also adopted to test our algorithms.

**5.2. Algorithm Evaluation.** To verify the superiority of the proposed algorithm, RMSE for regression problems and testing accuracy for classification problems are, respectively, computed. The initial size of the dataset is very small, which equals to the number of the hidden nodes to guarantee the model to work. All the data then is sent to the model in a one-by-one learning mode. The evaluation results are presented in Tables 3, 4, 5, and 6, which are, respectively, corresponding to the models with sigmoid hidden nodes and RBF hidden nodes for both regression and classification problems. Each result is an average of 50 trials. And in every trial of one problem, the training and testing samples were randomly adopted from the dataset that was addressed currently.

From the comparison results of four tables, we can easily find that EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN) are more time consuming than OS-ELM, but they still keep relatively fast speed at most of the time. It should be noted that the complexity of SEOS-ELM is adjustable, which depends on the threshold  $\lambda$ .

What is important, EOS-ELM, SEOS-ELM (GASEN), and SEOS-ELM (PSOSEN) all attain lower testing deviation and more accurate regression or classification results than OS-ELM, which shows the advantage of ensemble learning. In addition, both SEOS-ELM (GASEN) and SEOS-ELM (PSOSEN) are more accurate than EOS-ELM. This verifies that selective ensemble is better than simple ensemble method.

In terms of the comparison between SEOS-ELM (GASEN) and SEOS-ELM (PSOSEN), it can be observed that both of the two selective ensemble algorithms achieve comparable accuracy. However, the advantage of the new algorithm is that it is more computational efficient. This verifies that PSOSEN is a fast and accurate selective ensemble algorithm.

As an online learning algorithm, the online learning ability is another important evaluation criterion. To illustrate the online learning ability of the proposed algorithm, a simulated regression dataset is adopted. The dataset was generated from the function  $y = x^2 + 3x + 2$ , comprising 4500 training data and 1000 testing data. Noting that this function is chosen arbitrarily just to simulate the regression problem, Figures 3 and 4 explicitly depict the variability of training accuracy of SEOS-ELM (PSOSEN), EOS-ELM, and OS-ELM with respect to the number of training data in the process of learning. It can be observed that with the increasing number of training samples, RMSE values of the three methods significantly decline. As the online learning progressed, the training models are continuously updated and corrected. We can then conclude that the more training data the system learns, the more precise the model is. Whether sigmoid or RBF the hidden nodes is, SEOS-ELM always obtains smaller RMSE than EOS-ELM and OS-ELM, which indicates that the performance of SEOS-ELM is considerably accurate compared with the other methods. Moreover, the smaller testing deviation of SEOS-ELM in

TABLE 3: Comparison of algorithms for regression problems with sigmoid hidden nodes.

| Datasets           | Algorithm         | Number of nodes | Number of networks | Training time (s) | RMSE or Accuracy |              | Testing dev. |
|--------------------|-------------------|-----------------|--------------------|-------------------|------------------|--------------|--------------|
|                    |                   |                 |                    |                   | Training RMSE    | Testing RMSE |              |
| Abalone            | OS-ELM            | 25              |                    | 0.1191            | 0.0758           | 0.0782       | 0.0049       |
|                    | EOS-ELM           | 25              | 5                  | 0.5942            | 0.0754           | 0.0775       | 0.0023       |
|                    | SEOS-ELM (GASEN)  | 25              | 5                  | 7.3864            | 0.0744           | 0.0760       | 0.0016       |
|                    | SEOS-ELM (PSOSEN) | 25              | 5                  | 4.1528            | 0.0742           | 0.0758       | 0.0015       |
| Mackey-Glass       | OS-ELM            | 120             |                    | 0.9827            | 0.0177           | 0.0185       | 0.0018       |
|                    | EOS-ELM           | 120             | 5                  | 4.8062            | 0.0176           | 0.0183       | 0.0007       |
|                    | SEOS-ELM (GASEN)  | 120             | 5                  | 37.4371           | 0.0172           | 0.0180       | 0.0005       |
|                    | SEOS-ELM (PSOSEN) | 120             | 5                  | 25.1608           | 0.0173           | 0.0179       | 0.0006       |
| California Housing | OS-ELM            | 50              |                    | 0.6871            | 0.1276           | 0.1335       | 0.0035       |
|                    | EOS-ELM           | 50              | 5                  | 3.2356            | 0.1280           | 0.1337       | 0.0019       |
|                    | SEOS-ELM (GASEN)  | 50              | 5                  | 20.7635           | 0.1242           | 0.1321       | 0.0014       |
|                    | SEOS-ELM (PSOSEN) | 50              | 5                  | 15.6326           | 0.1238           | 0.1323       | 0.0014       |

TABLE 4: Comparison of algorithms for classification problems with sigmoid hidden nodes.

| Datasets           | Algorithm         | Number of nodes | Number of networks | Training time (s) | RMSE or Accuracy |              | Testing dev. |
|--------------------|-------------------|-----------------|--------------------|-------------------|------------------|--------------|--------------|
|                    |                   |                 |                    |                   | Training RMSE    | Testing RMSE |              |
| New-thyroid        | OS-ELM            | 20              |                    | 0.0043            | 93.18%           | 89.66%       | 0.1138       |
|                    | EOS-ELM           | 20              | 15                 | 0.0627            | 94.32%           | 90.92%       | 0.0276       |
|                    | SEOS-ELM (GASEN)  | 20              | 15                 | 1.2476            | 95.14%           | 91.58%       | 0.0201       |
|                    | SEOS-ELM (PSOSEN) | 20              | 15                 | 0.5012            | 95.23%           | 91.78%       | 0.0198       |
| Image segmentation | OS-ELM            | 180             |                    | 1.8432            | 97.07%           | 94.83%       | 0.0078       |
|                    | EOS-ELM           | 180             | 20                 | 36.2458           | 97.08%           | 94.79%       | 0.0055       |
|                    | SEOS-ELM (GASEN)  | 180             | 20                 | 432.1987          | 97.60%           | 95.12%       | 0.0045       |
|                    | SEOS-ELM (PSOSEN) | 180             | 20                 | 254.0721          | 97.56%           | 95.21%       | 0.0043       |
| Satellite image    | OS-ELM            | 400             |                    | 42.2503           | 92.82%           | 88.92%       | 0.0058       |
|                    | EOS-ELM           | 400             | 20                 | 853.2675          | 92.80%           | 89.05%       | 0.0026       |
|                    | SEOS-ELM (GASEN)  | 400             | 20                 | 8241.4093         | 93.54%           | 89.92%       | 0.0017       |
|                    | SEOS-ELM (PSOSEN) | 400             | 20                 | 6928.0968         | 93.96%           | 90.16%       | 0.0018       |

TABLE 5: Comparison of algorithms for regression problems with RBF hidden nodes.

| Datasets           | Algorithm         | Number of nodes | Number of networks | Training time (s) | RMSE or Accuracy |              | Testing dev. |
|--------------------|-------------------|-----------------|--------------------|-------------------|------------------|--------------|--------------|
|                    |                   |                 |                    |                   | Training RMSE    | Testing RMSE |              |
| Abalone            | OS-ELM            | 25              |                    | 0.3445            | 0.0753           | 0.0775       | 0.0027       |
|                    | EOS-ELM           | 25              | 25                 | 8.5762            | 0.0752           | 0.0773       | 0.0023       |
|                    | SEOS-ELM (GASEN)  | 25              | 25                 | 54.2453           | 0.0742           | 0.0760       | 0.0016       |
|                    | SEOS-ELM (PSOSEN) | 25              | 25                 | 49.3562           | 0.0741           | 0.0761       | 0.0017       |
| Mackey-Glass       | OS-ELM            | 120             |                    | 1.6854            | 0.0181           | 0.0185       | 0.0092       |
|                    | EOS-ELM           | 120             | 5                  | 8.4304            | 0.0171           | 0.0171       | 0.0028       |
|                    | SEOS-ELM (GASEN)  | 120             | 5                  | 79.3216           | 0.0155           | 0.0158       | 0.0021       |
|                    | SEOS-ELM (PSOSEN) | 120             | 5                  | 55.1469           | 0.0159           | 0.0156       | 0.0016       |
| California Housing | OS-ELM            | 50              |                    | 1.8329            | 0.1298           | 0.1317       | 0.0017       |
|                    | EOS-ELM           | 50              | 5                  | 9.0726            | 0.1296           | 0.1316       | 0.0011       |
|                    | SEOS-ELM (GASEN)  | 50              | 5                  | 69.8636           | 0.1216           | 0.1262       | 0.0008       |
|                    | SEOS-ELM (PSOSEN) | 50              | 5                  | 64.9625           | 0.1202           | 0.1243       | 0.0009       |

TABLE 6: Comparison of algorithms for classification problems with RBF hidden nodes.

| Datasets           | Algorithm         | Number of nodes | Number of networks | Training time (s) | RMSE or Accuracy | Testing dev. |
|--------------------|-------------------|-----------------|--------------------|-------------------|------------------|--------------|
| New-thyroid        | OS-ELM            | 20              |                    | 0.0118            | 93.45% 89.92%    | 0.0702       |
|                    | EOS-ELM           | 20              | 15                 | 0.1682            | 93.87% 89.86%    | 0.0428       |
|                    | SEOS-ELM (GASEN)  | 20              | 15                 | 1.9745            | 94.53% 91.05%    | 0.0332       |
|                    | SEOS-ELM (PSOSEN) | 20              | 15                 | 1.2315            | 94.68% 91.32%    | 0.0315       |
| Image segmentation | OS-ELM            | 180             |                    | 2.6702            | 94.98% 91.92%    | 0.0324       |
|                    | EOS-ELM           | 180             | 5                  | 13.2174           | 94.39% 91.35%    | 0.0148       |
|                    | SEOS-ELM (GASEN)  | 180             | 5                  | 128.3215          | 95.62% 95.06%    | 0.0085       |
|                    | SEOS-ELM (PSOSEN) | 180             | 5                  | 90.2856           | 96.02% 95.24%    | 0.0079       |
| Satellite image    | OS-ELM            | 400             |                    | 45.2702           | 93.62% 89.54%    | 0.0056       |
|                    | EOS-ELM           | 400             | 10                 | 448.1347          | 93.86% 89.37%    | 0.0034       |
|                    | SEOS-ELM (GASEN)  | 400             | 10                 | 4263.1406         | 94.61% 90.38%    | 0.0022       |
|                    | SEOS-ELM (PSOSEN) | 400             | 10                 | 3145.8528         | 94.85% 90.57%    | 0.0019       |

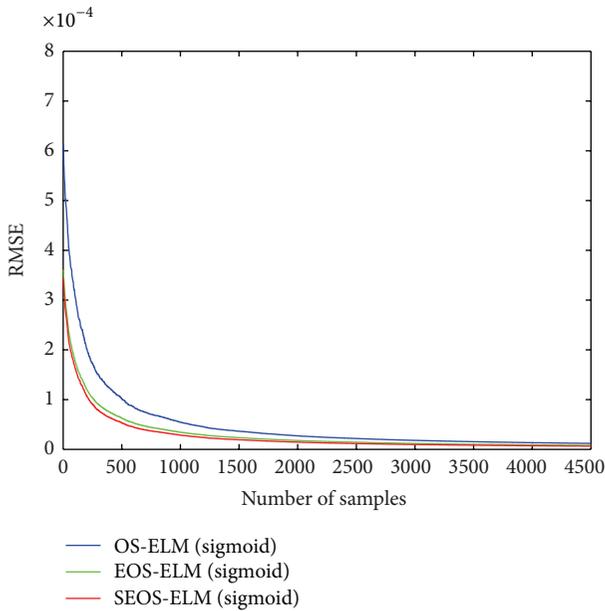


FIGURE 3: RMSE with respect to the number of training samples for sigmoid hidden nodes.

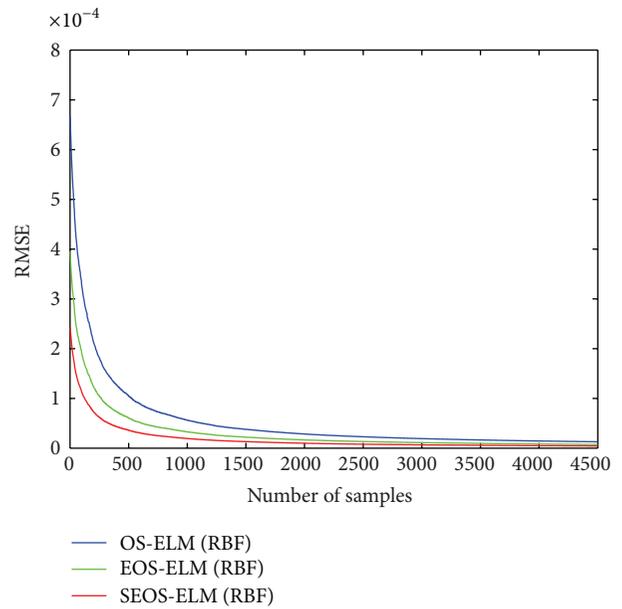


FIGURE 4: RMSE with respect to the number of training samples for RBF hidden nodes.

Table 3 to Table 6 also confirms the stability performance of SEOS-ELM.

### 6. Discussion

In the experiments, PSOSEN showed its higher accuracy than the original OS-ELM and simple ensemble of OS-ELM, which verified the feasibility of the selective ensemble method. In addition, compared with GASEN, PSOSEN showed comparable accuracy while much faster learning speed. Taking the complexity and accuracy into consideration, PSOSEN is a good choice for selective ensemble. Experiments on online version ELM have demonstrated the advantages. However, it should be noted that, as a general selective ensemble method, PSOSEN is applicable to any learning algorithms, both batch

learning and online learning. So, applying PSOSEN to other learning algorithms are of interest in the future.

The experiments also showed that although ensemble learning, both simple ensemble and selective ensemble, attains higher accuracy, it is more time consuming than the original learning algorithm. In addition, selective ensemble is slower than simple ensemble. As a selective ensemble method, PSOSEN is also slower than the original learning algorithm and the simple ensemble. So, selective ensemble is a trade-off between complexity and accuracy. In the future, new selective ensemble method should be designed to further improve the speed of the algorithm.

### 7. Conclusion

In this paper, PSOSEN is proposed as a novel selective ensemble algorithm. Benefiting from the fast speed of PSO,

PSOSEN proves to be faster than other selective ensemble algorithms. It is a general selective ensemble algorithm, which is applicable to any learning algorithms. To improve the generalization performance of the online learning algorithm, we then apply PSOSEN to OS-ELM. And in purpose of balancing the complexity and accuracy, an adaptive selective ensemble framework for OS-ELM is designed. Experiments were carried out on UCI data set. The results convincingly show that the new algorithm improves the generalization performance of OS-ELM and also keeps balance on complexity.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgment

This work is partially supported by Natural Science Foundation of China (41176076, 31202036, 51379198, and 51075377).

### References

- [1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Neural Networks, 2nd edition, 2004.
- [2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [3] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [4] R. Zhang, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 3, pp. 485–495, 2007.
- [5] Y. Xu, Z. Y. Dong, J. H. Zhao, P. Zhang, and K. P. Wong, "A reliable intelligent system for real-time dynamic security assessment of power systems," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1253–1263, 2012.
- [6] K. Choi, K. A. Toh, and H. Byun, "Incremental face recognition for large-scale social network services," *Pattern Recognition*, vol. 45, no. 8, pp. 2868–2883, 2012.
- [7] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [8] J. C. Yin, Z. J. Zou, F. Xu, and N. N. Wang, "Online ship roll motion prediction based on grey sequential extreme learning machine," *Neurocomputing*, vol. 129, pp. 168–174, 2014.
- [9] X. Wang and M. Han, "Online sequential extreme learning machine with kernels for nonstationary time series prediction," *Neurocomputing*, vol. 145, pp. 90–97, 2014.
- [10] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, June 2014.
- [11] N. Liu and H. Wang, "Ensemble based extreme learning machine," *IEEE Signal Processing Letters*, vol. 17, no. 8, pp. 754–757, 2010.
- [12] X. Xue, M. Yao, Z. Wu, and J. Yang, "Genetic ensemble of extreme learning machine," *Neurocomputing*, vol. 129, pp. 175–184, 2014.
- [13] H. J. Rong, Y. S. Ong, A. H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1–3, pp. 359–366, 2008.
- [14] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.
- [15] P. S. A. Krogh, "Learning with ensembles: how over-fitting can be useful," in *Proceedings of the 1995 Neural Information Processing Systems Conference*, vol. 8, p. 190, 1996.
- [16] Y. Lan, Y. C. Soh, and G. B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13–15, pp. 3391–3395, 2009.
- [17] Z. H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, no. 1–2, pp. 239–263, 2002.
- [18] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, New York, NY, USA, 2010.
- [19] C. Blake and C. J. Merz, "UCI repository of machine learning databases," Tech. Rep., University of California, 1998.
- [20] J. Kennedy and W. M. Spears, "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 78–83, Anchorage, Alaska, USA, May 1998.

## Research Article

# One-Class Classification with Extreme Learning Machine

Qian Leng,<sup>1</sup> Honggang Qi,<sup>1</sup> Jun Miao,<sup>2</sup> Wentao Zhu,<sup>2</sup> and Guiping Su<sup>1</sup>

<sup>1</sup>School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 101408, China

<sup>2</sup>Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China

Correspondence should be addressed to Jun Miao; [jmiao@ict.ac.cn](mailto:jmiao@ict.ac.cn)

Received 13 August 2014; Revised 8 November 2014; Accepted 10 November 2014

Academic Editor: Zhan-li Sun

Copyright © 2015 Qian Leng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

One-class classification problem has been investigated thoroughly for past decades. Among one of the most effective neural network approaches for one-class classification, autoencoder has been successfully applied for many applications. However, this classifier relies on traditional learning algorithms such as backpropagation to train the network, which is quite time-consuming. To tackle the slow learning speed in autoencoder neural network, we propose a simple and efficient one-class classifier based on extreme learning machine (ELM). The essence of ELM is that the hidden layer need not be tuned and the output weights can be analytically determined, which leads to much faster learning speed. The experimental evaluation conducted on several real-world benchmarks shows that the ELM based one-class classifier can learn hundreds of times faster than autoencoder and it is competitive over a variety of one-class classification methods.

## 1. Introduction

One-class classification [1, 2] has received much interest during recent years, which has also been known as novelty or outlier detection. Different from normal classification, data samples from only one class, called the target class, are well characterized, while there are no or few samples from the other class (also called the outlier class). To reveal the necessity of one-class classification, we take the case of online shopping service as an example. In order to recommend goods users want, it is convenient to track the users' history shopping lists (positive training samples), while collection of negative training samples is challenging because it is hard to say which one users dislike. Other applications include machine fault detection [3], disease detection [4], and credit scoring [5]. The goal is to "teach" the classifier through observing target samples so that it can be applied to select unknown samples similar to the target class and reject samples which deviate significantly from the target class.

Various types of one-class classifier have been designed and applied in different fields; see [6] for a comprehensive review. Early attempt to obtain a one-class classifier is by estimating the probability density functions based on training

data. Parzen density estimation [7, 8] superposes kernel functions on individual training samples to estimate the probability density function. Naive Parzen density estimation, similar to Naive Bayes approach used for classification, fits a Parzen density estimation on each individual feature and multiplies the results for final density estimation. A test sample is rejected if its estimated probability is below a threshold. However, estimating the true density distribution usually requires a large number of training samples.

A simpler task is to find the domain of the data distribution. Schölkopf et al. [9] constructed a hyperplane which is maximally distant from the origin to separate the regions that contain no data. An alternative approach is to find a hypersphere [10] instead of a hyperplane to include the most target data with the minimum radius. Both approaches are cast out in the form of quadratic programming, while some approaches [11–13] are of linear programming. One-class LP classifier [11] minimizes the volume of the prism, which is cut by a hyperplane that bounds the data from above with some mild constraints on dissimilarity representations. Lanckriet et al. [13] propose the one-class minimax probability machine that minimizes the worst case probability of misclassification of test data, using only the mean and covariance matrix

of the target distribution. When kernel methods are used, the aforementioned domain-based classifiers [2] can obtain more flexible descriptions. Recently, a minimum spanning tree based one-class classifier [14] was proposed. It considered graph edges as additional set of virtual target objects. By constructing a minimum spanning tree, recognition of a new test sample is determined by the shortest distance to the closest edge of that tree.

Autoencoder neural network is one of the reconstruction methods [1] to build a one-class classifier. The simplest architecture of such model is based on the single-hidden layer feed-forward neural networks (SLFNs). Usually, the hidden layer contains a smaller number of nodes than the number of input nodes which works like an information bottleneck. The classifier reproduces the input patterns at the output layer through minimizing the reconstruction error. However, standard backpropagation (BP) algorithm is used to train the networks, which is quite time-consuming. Extreme learning machine [15, 16] is originally developed to address the slow learning speed problem of gradient based learning algorithms for its iterative tuning of the networks' parameters. It randomly selects all parameters of the hidden neurons and analytically determines the output weights. It is stated [17, 18] in theory that ELM tends to provide the best generalization performance at extreme learning speed since it is a simple tuning-free algorithm.

In this paper, the proposed one-class classifier based on ELM is constructed for situations where only the target class is well described. The proposed one-class classifier utilizes the unified ELM learning theory [17], which leads to extreme learning speed and superior generalization performance. Moreover, the classifier further lessens the human intervention since it is not limited to specific target labels. Both random feature mappings and kernels can be adopted for such classifier which makes it more flexible to unique target descriptions. Constructing the proposed classifier for three quite different specific-designed artificial datasets demonstrates the classifier's ability to describe universal target class distributions. When real-world datasets are evaluated, the proposed one-classifier is competitive over a variety of one-class models and learns hundreds of times faster than autoencoder neural network for one-class classification.

The rest of the paper is organized as follows. Section 2 briefly reviews extreme learning machine. In Section 3, we first describe the hypersphere perceptron as a one-class classifier and then introduce our proposed ELM based one-class classifier. Section 4 describes the experiments conducted on both artificial and real-world datasets. Finally, Section 5 presents the conclusion of the work.

## 2. Brief Review of ELM

ELM aims to reach not only the smallest training error but also the smallest norm of output weights [16] between the hidden layer and the output layer. According to Bartlett's theory [19], the smaller norm of weights is, the better generalization performance of networks tends to have. Thus, better generalization performance can be expected for ELM

networks. In [17], equality constraints are used in ELM, which provides a unified solution for regression, binary, and multiclass classifications.

*2.1. Equality-Optimization-Constraints-Based ELM.* Given  $N$  training data  $(\mathbf{x}_i, t_i)_{i=1}^N$ , where  $\mathbf{x}_i = [x_{i1}, \dots, x_{in}]^T \in R^n$  is the individual feature vector with dimension  $n$  and  $t_i \in R^m$  is the desired target output, in the one-class classification case, single output node ( $m = 1$ ) is enough. The ELM output function can be formulated as

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = \sum_{j=1}^L \beta_j G(\mathbf{w}_j, b_j, \mathbf{x}), \quad (1)$$

where  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_L]^T$  is the vector of the output weights between the hidden layer and the output layer,  $\mathbf{w}_j = [w_{j1}, \dots, w_{jn}]^T$  is the input weights connecting input nodes with the  $j$ th hidden node,  $b_j$  is the bias of the  $j$ th hidden node,  $\mathbf{h}(\mathbf{x}) = [G(\mathbf{w}_1, b_1, \mathbf{x}), \dots, G(\mathbf{w}_L, b_L, \mathbf{x})]^T$  is the output vector of the hidden layer with respect to input  $\mathbf{x}$ , and  $G(\mathbf{w}, b, \mathbf{x})$  is the activation function (e.g., sigmoid function  $G(\mathbf{w}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{w}^T \cdot \mathbf{x} + b)))$ ) satisfying ELM universal approximation capability theorems [20, 21]. In fact,  $\mathbf{h}(\mathbf{x})$  is a known nonlinear feature mapping which maps the training data  $\mathbf{x}$  from the  $n$ -dimensional input space to the  $L$ -dimensional ELM feature space [17]. The goal of ELM is to minimize the norm of output weights as well as the training errors, which is equivalent to

$$\begin{aligned} \min \quad & L_{P_{\text{ELM}}} = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} = t_i - \xi_i, \quad i = 1, \dots, N, \end{aligned} \quad (2)$$

where  $\xi_i$  is the slack variable of the training sample  $\mathbf{x}_i$  and  $C$  controls the tradeoff between the output weights and the errors. Based on the Karush-Kuhn-Tucker (KKT) theorem [22], the corresponding Lagrange function of the primal ELM optimization (2) is

$$L_{D_{\text{ELM}}} = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 - \sum_{i=1}^N \alpha_i (\mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} - t_i + \xi_i); \quad (3)$$

the following optimality conditions of (3) should be satisfied:

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \boldsymbol{\beta}} = 0 \implies \boldsymbol{\beta} = \sum_{i=1}^N \alpha_i, \quad \mathbf{h}(\mathbf{x}_i) = \mathbf{H}^T \boldsymbol{\alpha}, \quad (4a)$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \xi_i} = 0 \implies \alpha_i = C \xi_i, \quad i = 1, \dots, N, \quad (4b)$$

$$\frac{\partial L_{D_{\text{ELM}}}}{\partial \alpha_i} = 0 \implies \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} - t_i + \xi_i = 0, \quad i = 1, \dots, N, \quad (4c)$$

where  $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_N)]^T$  is the hidden layer output matrix and  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^T$  is the vector of Lagrange variables. Substituting (4a) and (4b) into (4c) we have

$$\left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right) \boldsymbol{\alpha} = \mathbf{T}. \quad (5)$$

Here  $\mathbf{I}$  is the identity matrix and  $\mathbf{T} = [t_1, \dots, t_N]^T$ . Substituting (5) into (4a), we get

$$\boldsymbol{\beta} = \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (6)$$

The ELM output function (1) can be further derived as

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = \mathbf{h}(\mathbf{x})^T \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (7)$$

If the hidden nodes' feature mapping  $\mathbf{h}(\mathbf{x})$  is unknown to users, kernel methods that satisfy Mercer's condition can be adopted:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j)$ . The ELM kernel output function can be written as

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta} = \mathbf{h}(\mathbf{x})^T \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ &= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left( \frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\text{ELM}} \right)^{-1} \mathbf{T} \end{aligned} \quad (8a)$$

and the kernel matrix for ELM is

$$\boldsymbol{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^T : \Omega_{\text{ELM}i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \quad (8b)$$

**2.2. Advances of ELM.** Extreme learning machine has gained much more popularity since its advent. It has been able to avoid the problem of time complexity which classic learning techniques are confronted with while providing better generalization performance with less human intervention. Because of such attractive features, researchers have extended the basic ELM to several different directions and many variants of ELM have been developed. For instance, online sequential ELM (OS-ELM) [23, 24] can learn the sequential coming data (one by one or chunk by chunk) with a small effort to update the output weights. The training data are discarded after being learned by the network and the output weights need not be retrained, which is especially efficient for time-series problems. Other typical works include fully complex ELM [25, 26], incremental ELM (I-ELM) [20, 21], sparse ELM [27], ELM with elastic output [28, 29], and ELM ensembles [30–32]. See [33] for further details on the many ELM variants.

When uncertainty is present in the dataset, integration of fuzzy logic system and extreme learning machine tends to enhance the generalization capability of ELM. In [34], a neurofuzzy Takagi-Sugeno-Kang (TSK) fuzzy inference system is constructed utilizing extreme learning machine. The number of inference rules is previously determined by the  $k$ -means method. One ELM is used to obtain the membership of each fuzzy rule and multiple ELM are used to obtain the consequent part. Rong et al. [35] show that type-1 fuzzy inference system (type-1 FLS) is equivalent to a generalized SLFN. Hence, the hidden nodes work as the antecedent part and the output weights as the consequent part. Then, extreme learning machine is directly applied to the type-1 FLS and the corresponding online sequential fuzzy ELM has also

been developed. Deng et al. [36] further extend the idea to type-2 fuzzy inference system (type-2 FLS) because of type-2 FLS's superiority in modeling high level uncertainty. With the most widely used interval type-2 FLS, the parameters of the antecedents are randomly initialized according to the ELM mechanism. The Moore-Penrose generalized inverse is used to initialize the parameters of the consequents and the parameters are finally refined by Karnik-Mendel algorithm [37]. Many applications have also been investigated in the literature. For example, the hybrid model of ELM with interval type-2 FLS has been applied for permeability prediction [38].

### 3. The Proposed One-Class Classifier

**3.1. Support Vector Data Description.** For a better understanding of one-class classifiers, support vector data description (SVDD) [10] is discussed here for one-class classification process. SVDD defines a spherically shaped boundary around the complete target set and is intuitively appealing since it regards the target class as a self-closed system. Let  $X = \{\mathbf{x}_i, i = 1, \dots, N\}$  be the training set, and  $\mathbf{x}_i \in R^n$  is drawn from the target distribution. SVDD aims to minimize the volume of the sphere as well as the training errors  $\xi_i$  for objects falling outside the boundary, which is equivalent to

$$\begin{aligned} \min \quad & L_{\text{SVDD}} = R^2 + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & \|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad i = 1, \dots, N \\ & \xi_i \geq 0, \quad \forall i, \end{aligned} \quad (9)$$

where  $R$  and  $\mathbf{a}$  are the hypersphere's radius and center, respectively. Parameter  $C$  controls the tradeoff between the volume and the errors.

The corresponding function of the primal SVDD optimization (9) is

$$\begin{aligned} L_{\text{SVDD}} &= R^2 + C \sum_{i=1}^N \xi_i \\ &= R^2 + C \sum_{i=1}^N \xi_i \\ &\quad - \sum_{i=1}^N \alpha_i \left( R^2 + \xi_i - (\|\mathbf{x}_i\|^2 - 2\mathbf{a} \cdot \mathbf{x}_i + \|\mathbf{a}\|^2) \right) - \sum_{i=1}^N \beta_i \xi_i \end{aligned} \quad (10)$$

with the Lagrange variables  $\alpha_i \geq 0$  and  $\beta_i \geq 0$ .  $L_{\text{SVDD}}$  should be minimized with respect to  $R$ ,  $\mathbf{a}$ ,  $\xi_i$  and maximized with respect to  $\alpha_i$ ,  $\beta_i$ .

Based on the Karush-Kuhn-Tucker (KKT) theorem [22], to get the optimal solutions of (10), we should have

$$\frac{\partial L_{\text{SVDD}}}{\partial R} = 0 \implies \sum_{i=1}^N \alpha_i = 1, \quad (11a)$$

$$\frac{\partial L_{\text{SVDD}}}{\mathbf{a}} = 0 \implies \mathbf{a} = \sum_{i=1}^N \alpha_i \mathbf{x}_i, \quad (11b)$$

$$\frac{\partial L_{\text{SVDD}}}{\xi_i} = 0 \implies C = \alpha_i + \beta_i. \quad (11c)$$

From (11c) and  $\alpha_i \geq 0$  and  $\beta_i \geq 0$ ,  $\beta_i$  can be removed and  $\alpha_i$  can be further limited to the interval  $[0, C]$ :

$$0 \leq \alpha_i \leq C. \quad (12)$$

Substituting (11a)–(11c) into (10), the dual optimization function can be derived as

$$L_{D_{\text{SVDD}}} = \sum_{i=1}^N \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (13)$$

subject to constraints (11a) and (12). To constitute a flexible data description model, kernel function  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ , with an implicit feature mapping  $\phi$  of the data into a higher dimensional feature space, can be adopted to replace the inner product  $(\mathbf{x}_i \cdot \mathbf{x}_j)$ . In this case, the corresponding dual optimization function is changed to

$$L_{D_{\text{SVDD}}} = \sum_{i=1}^N \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j). \quad (14)$$

The KKT conditions of the target functions are

$$\begin{aligned} \alpha_i (R^2 + \xi_i - \|\mathbf{x}_i - \mathbf{a}\|^2) &= 0, \\ \beta_i \xi_i &= 0. \end{aligned} \quad (15)$$

The constraints have to be enforced and we have three cases as follows:

$$(1) \alpha_i = 0$$

$$\begin{aligned} \beta_i = C &\implies \xi_i = 0, \\ \alpha_i = 0 &\implies R^2 \geq \|\mathbf{x}_i - \mathbf{a}\|^2, \end{aligned} \quad (16)$$

$$(2) 0 < \alpha_i < C$$

$$\begin{aligned} \beta_i > 0 &\implies \xi_i = 0, \\ 0 < \alpha_i < C &\implies R^2 = \|\mathbf{x}_i - \mathbf{a}\|^2, \end{aligned} \quad (17)$$

$$(3) \alpha_i = C$$

$$\begin{aligned} \beta_i = 0 &\implies \xi_i \geq 0, \\ \alpha_i = C &\implies R^2 \leq \|\mathbf{x}_i - \mathbf{a}\|^2. \end{aligned} \quad (18)$$

Only a small ratio of objects with  $\alpha_i > 0$  are called the support vectors. The dual optimization functions (13) and (14) are standard Quadratic Programming (QP) problems and the Lagrange variables  $\alpha_i$  can be obtained using some optimization methods such as SMO algorithm [39]. To test

a new object  $\mathbf{z}$ , its distance to the center of the sphere is calculated. The classifier will accept the object if the distance is less than or equal to the radius:

$$\|\mathbf{z} - \mathbf{a}\|^2 = (\mathbf{z} \cdot \mathbf{z})$$

$$-2 \sum_{i=1}^N \alpha_i (\mathbf{z} \cdot \mathbf{x}_i) + \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \leq R^2. \quad (19)$$

In addition to the batch learning model of SVDD, incremental learning methods [40] of SVM are extended to SVDD algorithm. Yin et al. [28] show an online fault diagnosis process through a hybrid model of incremental SVDD (ISVDD) and ELM with incremental output structure (IOELM). They used the ISVDD to detect the unknown failure model, and the output nodes of IOELM are adaptively increased to recognize the new failure mode.

**3.2. The ELM Based One-Class Classifier.** When data only from the target class are available, the one-class classifier is trained to accept target objects and reject objects that deviate significantly from the target class. In the training phase, the one-class classifier, which defines a distance function  $d$  between the objects and the target class, takes in the training set  $X$  to build the classification model. In general, the classification model contains two important parameters to be determined: threshold  $\theta$  and modal parameter  $\lambda$ . A generic test sample  $\mathbf{z}$  is accepted by the classifier if  $d(\mathbf{z} | X, \lambda) < \theta$ .

In the training phase, not all the training samples are to be accepted by the one-class classifier due to the presence of outliers or noisy data contained in the training set. Otherwise, the trained classification model may generalize poor to unknown test set when the training set includes abnormal data samples. Usually, threshold  $\theta$  is determined such that a user-specified fraction  $\mu$  of training samples most deviant from the target class are rejected. For instance, if one is told five percent of training samples are mislabeled, setting  $\mu = 0.05$  makes the classifier more robust. Even when all the samples are correctly labeled, rejecting a small fraction of training samples helps the classifier to learn the most representative model from the training samples.

Any one-class classifier has model parameters which influence the model complexity (flexibility), for example, the number of hidden nodes in autoencoder neural networks or the tradeoff parameter  $C$  of SVDD. Minimizing the errors of both the target and outlier classes on a cross-validation set is no longer available since there is no data from the outlier class. Fortunately, several model selection criteria [2] have been proposed. Assuming the uniform distribution of the outlier class, consistency-based model selection [41] method is one of the most effective methods used to select the model parameters. The basic idea is that the complexity of the classifier can be increased as long as it still fits the target data. The more complex the model, the smaller the volume of the classifier in the object space and the less the probability of outlier objects falling inside the domain of the classifier. In practice, one can make an ordering of the potential model parameters such that the latter parameter

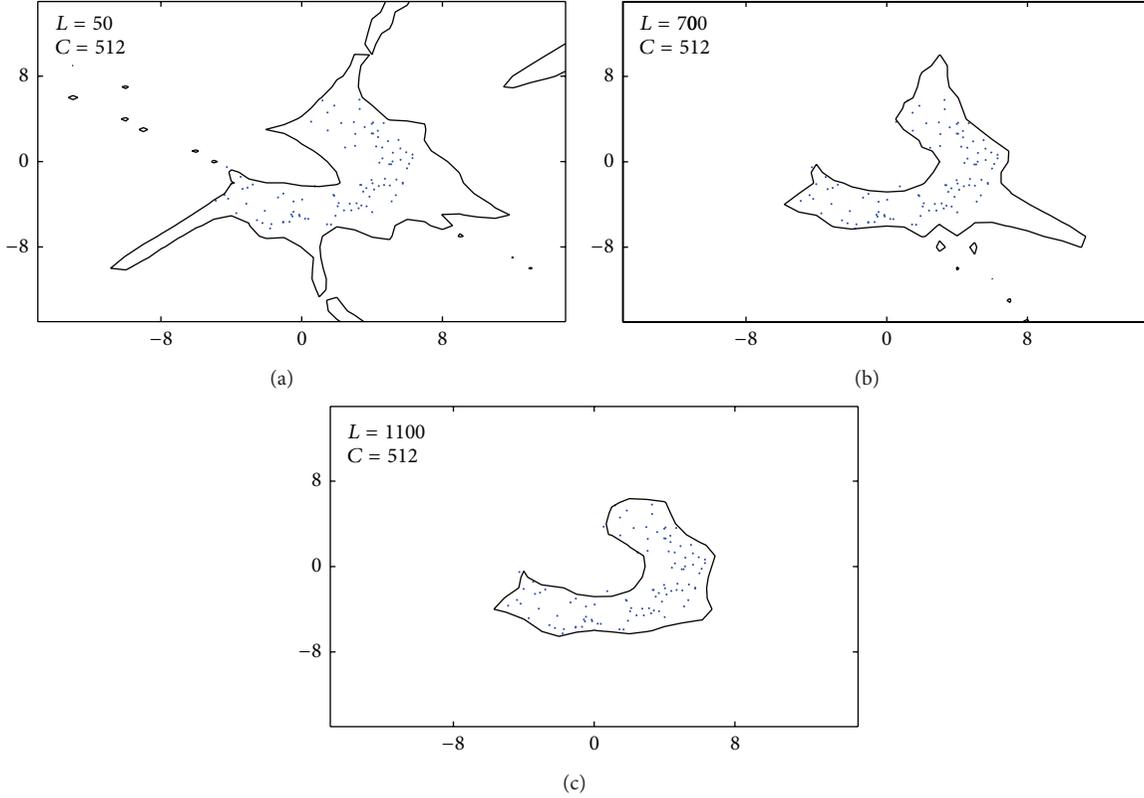


FIGURE 1: Example boundaries of the proposed classifier with different number of hidden nodes.

always yields the more complex classifier and chooses the most complex classifier without overfitting the target data.

The compactness hypothesis [42] is the basis for object recognition. It states that similar real world objects have to be close in the feature space. Therefore, for similar objects from the target class, the target outputs should be the same:

$$t_i = y, \quad \forall \mathbf{x}_i \in X, \quad (20)$$

where  $y$  is a real number. All the training samples' target outputs are set to the same value  $y$ . Then, the desired target output vector is  $\mathbf{T} = [t_1, \dots, t_N]^T = [y, \dots, y]^T$ . Training the samples from the target class can directly use the optimization function (2). For a new test sample  $\mathbf{z}$ , the distance function between the sample object and the target class is defined as

$$\begin{aligned} d_{\text{ELM}}(\mathbf{z} | X, \lambda) &= \left| \mathbf{h}(\mathbf{z})^T \boldsymbol{\beta} - y \right| \\ &= \left| \mathbf{h}(\mathbf{z})^T \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} - y \right|. \end{aligned} \quad (21)$$

The decision whether  $\mathbf{z}$  belongs to the target class or not is based on threshold  $\theta$ . Recall that  $\theta$  is optimized to reject a small fraction  $\mu$  of training samples to avoid overfitting. The distances of the training samples to the target class can be directly determined using (21) and the constraint of (2)

$$d_{\text{ELM}}(\mathbf{x}_i | X, \lambda) = \left| \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} - y \right| = |\xi_i|. \quad (22)$$

From (22), we find the distances are  $|\xi_i|$  and the larger  $|\xi_i|$  means the more deviant of the training sample  $\mathbf{x}_i$  from the target class. Hence, we derive threshold  $\theta$  based on a quantile function to reject the most deviant training samples. Denote the sorted sequence of the distances of training samples by  $\mathbf{d} = [d_{(1)}, \dots, d_{(N)}]$  such that  $d_{(1)} \geq \dots \geq d_{(N)}$ . Here,  $d_{(1)}$  and  $d_{(N)}$  represent the most and the least deviant samples. The function determining  $\theta$  can be written as

$$\theta = d_{\text{floor}(\mu \cdot N)}, \quad (23)$$

where  $\text{floor}(a)$  returns the largest integer not greater than  $a$ . Then, we can get the decision function for  $\mathbf{z}$  to the target class:

$$\begin{aligned} \mathcal{E}_{\text{ELM}}(\mathbf{z}) &= \text{sign}(\theta - d_{\text{ELM}}(\mathbf{z} | X, \lambda)) \\ &= \begin{cases} 1 & \mathbf{z} \text{ is classified as a target} \\ -1 & \mathbf{z} \text{ is classified as an outlier.} \end{cases} \end{aligned} \quad (24)$$

*Remark 1.* The target output  $y$  can be assigned to arbitrary real number except 0. When  $y = 0$ , seen from (6), the output weights between the hidden layer and the output layer become 0 ( $\boldsymbol{\beta} = \mathbf{0}$ ,  $\mathbf{0}$  is the  $L$ -dimensional zero vector). Therefore, the decision value of any sample  $\mathbf{z}$  is 0 using the proposed classifier. It is obvious that, in such case, the one-class classifier cannot distinguish between the target class and the outlier class. When  $y \neq 0$ , as there are infinite possible  $y$ , there seem to exist infinite ELM based one-class classifiers. To

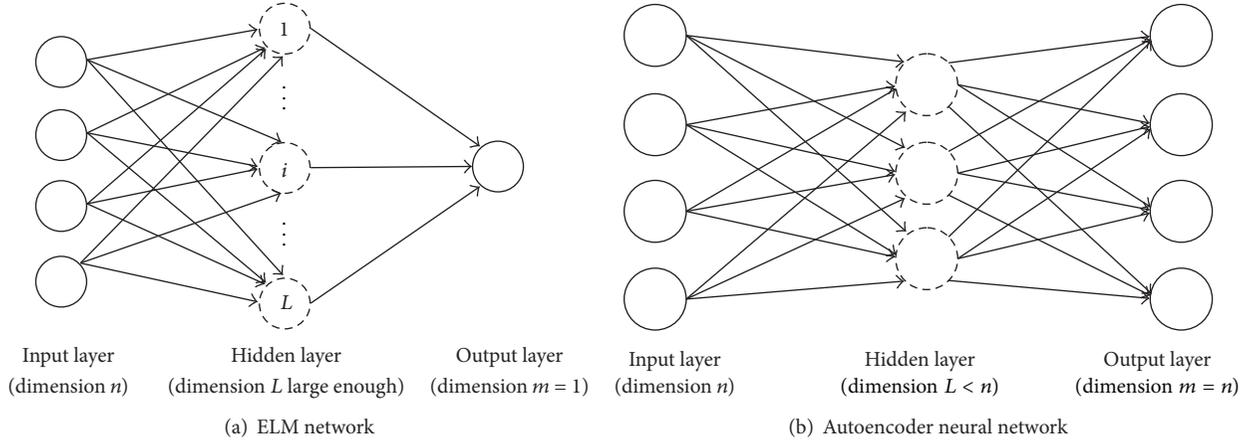


FIGURE 2: Comparisons between ELM network and autoencoder neural network: the number of hidden nodes in ELM network should be large enough according to ELM learning theory and one output node is enough for one-class classification, while the number of hidden nodes in autoencoder neural network is usually less than the feature dimension and the number of output nodes should be equal to the number of input nodes.

get a universal ELM based one-class classifier, we normalize the distance function (21) by dividing the target output  $y$

$$\begin{aligned}
 d_{\text{NORM-ELM}}(\mathbf{z} | X, \lambda) &= \left| \frac{\mathbf{h}(\mathbf{z}) \boldsymbol{\beta} - y}{y} \right| = \left| \frac{(\mathbf{h}(\mathbf{z}) \mathbf{H}^T (\mathbf{I}/C + \mathbf{H}\mathbf{H}^T)^{-1} \mathbf{T} - y)}{y} \right| \\
 &= \left| \frac{\mathbf{h}(\mathbf{z}) \mathbf{H}^T (\mathbf{I}/C + \mathbf{H}\mathbf{H}^T)^{-1} \mathbf{e}y - y}{y} \right| \\
 &= \left| \mathbf{h}(\mathbf{z})^T \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{e} - 1 \right|, \tag{25}
 \end{aligned}$$

where  $\mathbf{e}$  is the  $N$ -dimensional unit vector. The normalization formula (25) is to eliminate the possible bias introduced by the target output  $y$ . In practice, one can set the target output  $y = 1$  such that (21) is equivalent to (25) and the normalization step is implicitly done.

*Remark 2.* Both random feature mappings and kernels can be used for the proposed one-class classifier. When nonlinear piecewise continuous functions satisfying ELM universal approximation capability theorems [20, 21] are used as the activation function, the ELM network can approximate any target continuous function as long as the number of hidden nodes  $L$  is large enough. When the feature mapping is unknown, kernel methods can be adopted as shown in (8a) and (8b). Huang et al. [17] have shown ELM, the unified solution for regression, binary, and multiclass classifications. Since the same optimization formula (2) is used in the proposed one-class classifier, this paper also shows ELM, the unified learning mode for one-class classification.

Figure 1 shows the decision boundaries (black curves) of the classifier with incremental hidden nodes using sigmoid

TABLE 1: Specification of UCI datasets.

| Datasets       | Target class | # target | # outlier | # features |
|----------------|--------------|----------|-----------|------------|
| Spectf heart   | 0            | 55       | 212       | 44         |
| Arrhythmia     | Normal       | 245      | 207       | 279        |
| Sonar          | Mines        | 111      | 97        | 60         |
| Liver          | Healthy      | 145      | 200       | 6          |
| <i>E. coli</i> | Periplasm    | 52       | 284       | 7          |
| Diabetes       | Present      | 500      | 268       | 8          |
| Breast         | Benign       | 241      | 458       | 9          |
| Abalone        | Classes 1–8  | 1407     | 2770      | 8          |

function as the activation function. The dataset (blue points) is composed of 100 samples in the plane. Threshold  $\theta$  is determined such that  $\mu = 0.01$  and the model parameter  $C$  is automatically determined by the consistency-based model selection method. When the number of hidden nodes is small ( $L = 50$ ), the classifier fails to approximate the target region and some unexpected “holes” without any targets can be seen from the leftmost picture of Figure 1. The weakness alleviates as more hidden nodes are added. When the number of hidden nodes  $L$  gets large enough, the classifier can be close enough to describe the target class well. This is consistent with ELM universal approximation capability theorems [20, 21].

*Remark 3.* Autoencoder is one of the most effective neural networks approaches for one-class classification, which has been applied by Manevitz and Yousef for document retrieval [43]. Constrain the number of output nodes that must be equal to the number of input nodes ( $m = n$ ). The hidden layer in such a network actually acts as a bottleneck, where  $L < n$ . The idea is that while the bottleneck prevents learning the full identity function on  $n$ -space, the identity on the small set of examples is in fact learnable. Traditional learning algorithms like BP are used to train the network. Several challenging issues such as local minimum, trivial human intervention,

TABLE 2: The value of  $F_1$  measure with standard deviations (in parentheses) for a number of one-class classifiers. Twenty trials have been conducted for each dataset.

| Dataset<br>Classifier | SPECTF heart      | Arrhythmia        | $F_1$ | Sonar             | Liver             |
|-----------------------|-------------------|-------------------|-------|-------------------|-------------------|
| Naive Parzen          | 41.7 (4.2)        | 61.8 (1.1)        |       | 46.8 (2.2)        | 41.5 (0.9)        |
| Parzen                | 39.3 (1.7)        | 63.7 (1.2)        |       | 49.8 (2.9)        | 40.7 (1.4)        |
| $k$ -means            | 38.3 (4.7)        | 63.7 (1.7)        |       | 53.2 (3.2)        | 41.7 (1.4)        |
| 1-NN                  | 31.8 (2.6)        | 59.2 (1.5)        |       | <b>60.4 (2.2)</b> | 41.3 (1.3)        |
| $k$ -NN               | 34.7 (1.2)        | 62.4 (0.9)        |       | 55.3 (1.3)        | 42.0 (1.2)        |
| Autoencoder           | 39.3 (3.4)        | <b>64.8 (1.6)</b> |       | 50.6 (2.4)        | 42.2 (1.7)        |
| PCA                   | NaN <sup>1</sup>  | 26.3 (5.3)        |       | 37.2 (8.3)        | 41.1 (1.3)        |
| MST                   | 33.7 (1.7)        | 62.4 (0.8)        |       | 56.7 (1.8)        | 42.1 (1.1)        |
| $k$ -centers          | 36.4 (2.9)        | 62.8 (1.2)        |       | 53.3 (2.3)        | 41.6 (1.3)        |
| SVDD                  | 38.9 (4.7)        | 60.5 (4.8)        |       | 51.2 (5.8)        | 40.6 (3.1)        |
| MPM                   | 31.1 (8.7)        | 51.9 (5.0)        |       | 44.6 (6.3)        | 40.7 (2.0)        |
| LPDD                  | 38.3 (3.9)        | 63.8 (2.0)        |       | 52.2 (4.2)        | 40.7 (1.6)        |
| SVM                   | 38.1 (6.4)        | 63.4 (1.9)        |       | 53.6 (3.1)        | 40.5 (2.4)        |
| ELM                   | <b>42.6 (1.8)</b> | 63.6 (1.6)        |       | 54.2 (3.5)        | <b>43.0 (1.6)</b> |

<sup>1</sup>None of target data is recalled.

TABLE 3: The value of  $F_1$  measure with standard deviations (in parentheses) for a number of one-class classifiers. Twenty trials have been conducted for each dataset.

| Dataset<br>Classifier | <i>E. coli</i>    | Diabetes          | $F_1$ | Breast            | Abalone           |
|-----------------------|-------------------|-------------------|-------|-------------------|-------------------|
| Naive Parzen          | 71.7 (7.0)        | 68.7 (1.1)        |       | 82.4 (3.2)        | 51.8 (0.3)        |
| Parzen                | 75.1 (5.7)        | 67.7 (1.4)        |       | 80.1 (7.7)        | 49.0 (1.0)        |
| $k$ -means            | 54.6 (15.2)       | 68.9 (1.1)        |       | 58.8 (17.2)       | 45.2 (1.3)        |
| 1-NN                  | 21.2 (3.9)        | 64.8 (0.9)        |       | 35.3 (5.8)        | 35.7 (1.0)        |
| $k$ -NN               | 43.9 (14.2)       | 68.8 (1.0)        |       | 34.9 (7.5)        | 49.8 (1.4)        |
| Autoencoder           | 53.5 (15.8)       | 66.9 (1.3)        |       | 37.9 (10.9)       | 48.7 (2.7)        |
| PCA                   | 33.7 (15.4)       | 65.5 (1.9)        |       | 31.1 (1.0)        | 46.0 (0.5)        |
| MST                   | 36.3 (12.9)       | 67.5 (0.7)        |       | 34.4 (3.4)        | 47.3 (0.9)        |
| $k$ -centers          | 38.8 (6.5)        | 67.7 (1.1)        |       | 49.4 (22.9)       | 42.5 (2.6)        |
| SVDD                  | 50.9 (10.6)       | 61.1 (2.5)        |       | 68.0 (9.1)        | 44.5 (3.3)        |
| MPM                   | 38.8 (11.8)       | 63.5 (1.7)        |       | 71.7 (6.5)        | 44.9 (1.9)        |
| LPDD                  | 67.8 (11.0)       | 66.6 (0.8)        |       | 79.6 (7.5)        | 45.8 (2.0)        |
| SVM                   | 57.2 (12.8)       | 66.6 (1.1)        |       | <b>83.1 (3.0)</b> | 46.2 (1.2)        |
| ELM                   | <b>77.1 (4.8)</b> | <b>69.1 (1.1)</b> |       | 80.1 (5.1)        | <b>53.0 (0.7)</b> |

and time consuming in learning stage discourage people who are not familiar in the field to use it, while the ELM based one-class classifier can approximate the target class well as long as the dimensionality of the feature mappings is large enough (cf. Figure 2).

## 4. Experiments

*4.1. Artificial Datasets.* First, we illustrate the proposed method with both random feature mappings and kernels on three specific designed artificial datasets, which all contain 100 samples created in a 2D feature space. The first dataset

contains four Gaussian distributions (each has 25 samples) with the same unit covariance matrix but with different mean vectors. It is set to test the classifier's sensitivity to multimodality. The second dataset contains one Gaussian distribution with the first feature with a variance of 1 and the second feature with a variance of 40. Moreover, the two features are rotated over 45 degrees to construct a strong correlation. The third banana-shaped dataset, which has been shown in Section 3, contains one uniform distribution along an arc curve with some small position offsets. It is to test the influence of convexity. In Figure 3, the datasets (blue points) together with the decision boundaries (black curves) in the feature space are illustrated. Sigmoid function acts as

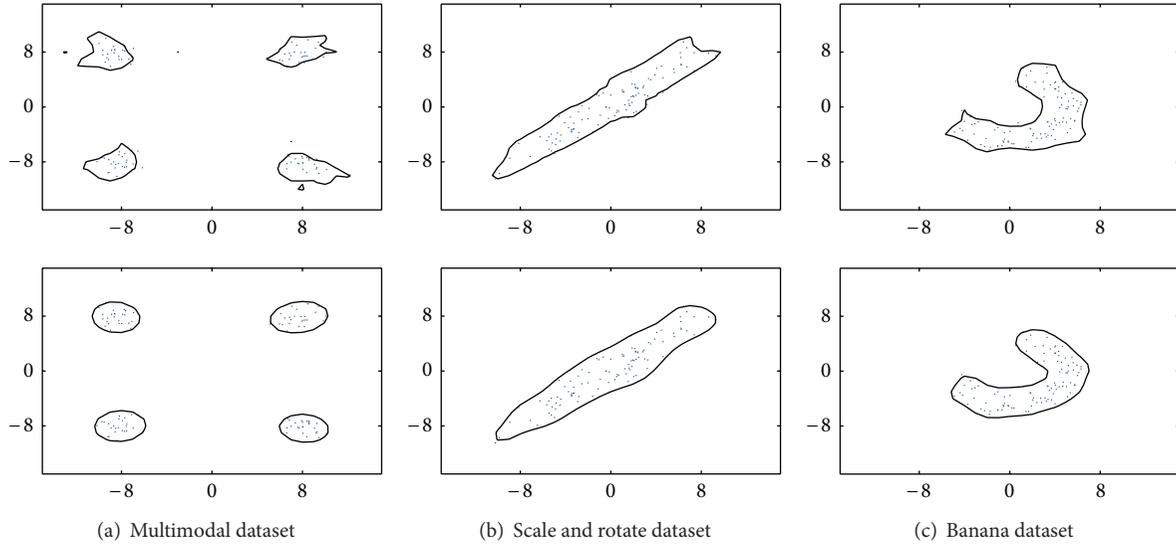


FIGURE 3: The upper row shows the boundaries of the method with random feature mappings and the bottom row shows the boundaries of the method with Gaussian kernel. Parameter  $C$  of the method with random feature mapping from left to right is  $2^{11}$ ,  $2^6$ , and  $2^9$ . Parameters  $(C, \sigma)$  of the method with Gaussian kernel from left to right are  $(2^2, 5.76)$ ,  $(2^0, 2.81)$ , and  $(2^{-4}, 1.55)$ .

the activation function for the method with random feature mappings ( $L$  large enough) and Gaussian kernel  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma^2)$  is used for the method with kernels. All the thresholds are determined such that  $\mu = 0.01$ . The pictures show that methods using both random feature mappings and kernels give reasonable results. However, the method with kernels tends to be superior to the method with random feature mappings since the boundary captures the distribution more precisely, while in Figure 3(a) some small “holes” still exist in the upper left and lower right regions for the method with random feature mappings.

**4.2. UCI Datasets.** This section compares the performance of the proposed method with a variety of one-class classification algorithms. The popular one-class classifiers to be compared include Parzen [7], Naive Parzen,  $k$ -means [44],  $k$ -centers [45], 1-NN [46],  $k$ -NN [47], autoencoder, PCA [48], MST [14], MPM [13], SVDD [10], LPDD [11], and SVM [9]. The implementations for one-class SVM are carried out using compiled C-coded SVM packages: LIBSVM [49]. All the other algorithms are conducted with Matlab toolbox DD\_TOOLS [50]. Binary and multiclass classification datasets taken from UCI Machine Learning Repository [51] are used. The specifications of the datasets are shown in Table 1. The datasets are transformed for one-class classification by setting a chosen class as the target class and all the other classes as the outlier class.

In our experiments, all the inputs have been normalized into range  $[0, 1]$ . The samples from the target class are equally partitioned in two sets for training and testing, respectively. All one-class classifiers are trained on target data only and tested on both the remaining target data and all other

TABLE 4: The value of precision and recall for two datasets (arrhythmia and *E. coli*).

| Dataset Classifier | Arrhythmia |        | <i>E. coli</i> |        |
|--------------------|------------|--------|----------------|--------|
|                    | Precision  | Recall | Precision      | Recall |
| Naive Parzen       | 45.6       | 96.0   | 63.7           | 86.2   |
| Parzen             | 52.0       | 82.4   | 71.5           | 80.6   |
| $k$ -means         | 52.5       | 81.0   | 52.4           | 64.6   |
| 1-NN               | 44.5       | 88.6   | 12.1           | 90.2   |
| $k$ -NN            | 47.8       | 90.2   | 31.9           | 87.3   |
| Autoencoder        | 52.6       | 84.7   | 47.3           | 73.5   |
| PCA                | 79.1       | 15.9   | 23.2           | 74.4   |
| MST                | 47.3       | 91.8   | 23.9           | 90.0   |
| $k$ -centers       | 50.4       | 83.4   | 29.3           | 68.9   |
| SVDD               | 55.7       | 69.3   | 48.8           | 66.9   |
| MPM                | 64.0       | 43.9   | 38.4           | 45.4   |
| LPDD               | 51.7       | 83.3   | 67.8           | 75.4   |
| SVM                | 52.4       | 80.5   | 57.8           | 65.8   |
| ELM                | 52.8       | 80.1   | 83.2           | 72.3   |

nontarget data. To assess the performance, we use  $F_1$  measure [52], which is defined as a combination of recall ( $R$ ) and precision ( $P$ ) with an equal weight in the following form:

$$F_1(R, P) = \frac{2RP}{(R + P)}. \quad (26)$$

All the thresholds  $\theta$  are determined such that  $\mu = 0.1$ . The Gaussian kernel is used in Parzen, Naive Parzen, MPM, SVDD, SVM, and ELM. The consistency-based model selection method is employed to select the model parameters. For

TABLE 5: Running time for ELM, autoencoder, and SVDD over twenty trials.

| Classifier<br>Dataset | ELM           |              | Autoencoder   |              | SVDD          |              |
|-----------------------|---------------|--------------|---------------|--------------|---------------|--------------|
|                       | Training time | Testing time | Training time | Testing time | Training time | Testing time |
| SPECTF heart          | 0.3           | 0.1          | 93.2          | 1.1          | 0.4           | 0.1          |
| Arrhythmia            | 0.2           | 0.2          | 17462.0       | 8.7          | 1.6           | 0.2          |
| Sonar                 | 0.1           | 0.1          | 248.8         | 1.4          | 0.7           | 0.1          |
| Liver                 | 0.1           | 0.1          | 9.9           | 0.9          | 1.3           | 0.2          |
| <i>E. coli</i>        | 0.1           | 0.1          | 8.2           | 0.9          | 0.7           | 0.2          |
| Diabetes              | 0.3           | 0.2          | 37.3          | 0.9          | 6.2           | 0.2          |
| Breast                | 0.1           | 0.2          | 45.3          | 0.9          | 2.5           | 0.3          |
| Abalone               | 1.0           | 2.5          | 64.2          | 1.0          | 118.0         | 2.5          |

Parzen, MPM, SVDD, SVM, and ELM, the kernel parameter  $\sigma$  is chosen from 20 aliquots between the minimum and maximum pairwise object distances, so as the smoothing parameter of sigmoid transform function used in LPDD. For  $k$ -means,  $k$ -centers, parameter  $k$  is selected from the range  $\{1, 2, \dots, 20\}$ . For ELM, another parameter  $C$  is chosen from the range  $\{2^{-24}, 2^{-23}, \dots, 2^{25}\}$  and we set  $\sigma$  a higher priority than  $C$ ; that is, when the parameter combinations,  $(\sigma_1, C_1)$  and  $(\sigma_2, C_2)$ , both obtain consistent boundaries, we always choose a smaller  $\sigma$  rather than a larger  $C$ . We try every possible parameter setting and find the most complex classifier as long as the classifier is consistent. For Naive Parzen and  $k$ -NN, the leave-one-out maximum likelihood estimation is used. One-class PCA retains 0.95 variance for the training set. For MST, the complete minimum spanning tree is used. The number of hidden nodes in autoencoder neural network is carefully chosen from a large range and the optimal number is selected.

All the experiments are carried out in Matlab R2013a environment running in E5504 2 GHz CPU, 4 GB RAM. Twenty trials have been conducted for each dataset and the average  $F_1$  and corresponding standard deviations are shown in Tables 2 and 3. The best results are shown in boldface. As an example, we give a detailed description for the diabetes experiment. First, all the samples from both the target class and the outlier class are normalized into range  $[0, 1]$ . Then, the 500 training target samples are randomly divided into two equal sets (250 samples for each set). One of the sets is used for training the one-class classifier and the other set, together with all the samples from the outlier class, is used for testing only. After that, the consistency-based model selection method is employed to select the model parameters for each classifier using only the training set. Finally, the other target set with the outlier set is judged by the trained classifier with precision and recall recorded.  $F_1$  value is then derived as (26). The same procedure repeats for twenty times and the corresponding mean and deviation values are calculated. It can be seen that the generalization performance of ELM is the best in five of the eight experiments while in the other experiments, except for sonar dataset, the performance is comparable to the best classifier. Table 4 presents a detailed performance comparison of two datasets, including precision and recall.

Table 5 reports the execution time comparisons in seconds between the ELM, autoencoder, and SVDD classifiers

for all the eight experiments. As observed from Table 5, the advantage of the ELM on training time is quite obvious. ELM can generally learn hundreds of times faster than autoencoder neural network due to the tuning-free mechanism. Besides, ELM also learns much faster than SVDD without solving a QP problem. For testing time, since autoencoder may obtain a more compact network and the parameters have been tuned in the training phase, the computational time depends on the specific task. The computational complexity of ELM mostly depends on the number of samples while autoencoder depends on both the number of samples and the number of dimensions. Thus, for datasets with relatively small size and high dimensions, such as arrhythmia dataset, ELM obtains a smaller testing time, while for datasets with relatively large size and low dimensions, such as abalone dataset, autoencoder reacts faster to the testing samples. However, ELM still tends to outperform autoencoder with respect to both training time and accuracy. It is obvious that ELM and SVDD obtain a similar testing time since both of them utilize a kernel function.

## 5. Conclusion

This paper presents a simple and efficient one-class classifier utilizing extreme learning machine, which also shows ELM, the unified learning mode for one-class classification. Both random feature mappings and kernels can be used for the proposed classifier while the method with kernels tends to be superior to the method with random feature mappings. Moreover, the proposed classifier with kernels achieves the best results on five of the eight UCI datasets, which suggests ELM being effective for one-class classification problem. We have also discussed the relationships and differences between autoencoder neural network and ELM network for one-class classification. Although autoencoder neural network has been successfully applied in many applications, the slow gradient based method is still used to tune all the parameters, which is far slower than required. On the other hand, the ELM based one-class classifier has an analytical solution which can obtain superior generalization performance at much faster learning speed. Possible future directions include the fusion of fuzzy logic and ELM for one-class classification, one-class classifier ensembles with ELM, and substituting autoencoder with the ELM based one-class classifier for deep learning.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is partially sponsored by Natural Science Foundation of China (nos. 61175115, 61272320, 61379100, and 61472388). The authors would like to thank the helpful discussions with Mr. Fan Wang and Dr. Laiyun Qing.

## References

- [1] D. M. J. Tax, *One-class classification [Ph.D. thesis]*, Delft University of Technology, 2001.
- [2] P. Juszczak, *Learning to recognise. A study on one-class classification and active learning [Ph.D. thesis]*, Delft University of Technology, Delft, Netherlands, 2006.
- [3] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395–408, 2005.
- [4] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, C. Pellegrini, and A. Geissbuhler, "An application of one-class support vector machines to nosocomial infection detection," in *Proceedings of Medical Informatics*, 2004.
- [5] K. Kennedy, B. Mac Namee, and S. J. Delany, "Credit scoring: solving the low default portfolio problem using one-class classification," in *Proceedings of the 20th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 168–177, 2009.
- [6] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.
- [7] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.
- [8] R. P. W. Duin, "On the choice of smoothing parameters for Parzen estimators of probability density functions," *IEEE Transactions on Computers*, vol. 25, no. 11, pp. 1175–1179, 1976.
- [9] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [10] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.
- [11] E. Pekalska, D. M. J. Tax, and R. P. W. Duin, "One-class LP classifier for dissimilarity representations," in *Neural Information Processing Systems*, pp. 761–768, MIT Press, Cambridge, Mass, USA, 2003.
- [12] C. Campbell and K. P. Bennett, "A linear programming approach to novelty detection," in *Neural Information Processing Systems*, pp. 395–401, 2000.
- [13] G. R. G. Lanckriet, L. El Ghaoui, and M. I. Jordan, "Robust novelty detection with single-class MPM," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and T. Obermayer, Eds., vol. 15, pp. 905–912, MIT Press, Cambridge, Mass, USA, 2003.
- [14] P. Juszczak, D. M. J. Tax, E. Pekalska, and R. P. W. Duin, "Minimum spanning tree based one-class classifier," *Neurocomputing*, vol. 72, no. 7–9, pp. 1859–1869, 2009.
- [15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '04)*, pp. 985–990, Budapest, Hungary, July 2004.
- [16] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [17] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [18] W. Zhu, J. Miao, and L. Qing, "Constrained extreme learning machine: a novel highly discriminative random feedforward neural network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '14)*, Beijing, China, June 2014.
- [19] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.
- [20] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [21] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.
- [22] R. Fletcher, *Practical Methods of Optimization: Volume 2 Constrained Optimization*, Wiley, New York, NY, USA, 1981.
- [23] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [24] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, vol. 34, no. 6, pp. 2284–2292, 2004.
- [25] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, no. 1–4, pp. 306–314, 2005.
- [26] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, no. 4–6, pp. 576–583, 2008.
- [27] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.
- [28] G. Yin, Y.-T. Zhang, Z.-N. Li, G.-Q. Ren, and H.-B. Fan, "Online fault diagnosis method based on incremental support vector data description and extreme learning machine with incremental output structure," *Neurocomputing*, vol. 128, pp. 224–231, 2014.
- [29] T. Wang, S. Wang, and H. Zhang, "Dynamic extreme learning machine: a learning algorithm for neural network with elastic output structure," in *Proceedings of the International Symposium on Intelligent Information Systems and Applications*, pp. 271–275, 2009.
- [30] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, "GPU-accelerated and parallelized ELM ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, 2011.

- [31] Y. Sun, Y. Yuan, and G. Wang, "An OS-ELM based distributed ensemble classification framework in P2P networks," *Neurocomputing*, vol. 74, no. 16, pp. 2438–2443, 2011.
- [32] Y. Lan, Y. C. Soh, and G.-B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13–15, pp. 3391–3395, 2009.
- [33] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [34] Z.-L. Sun, K.-F. Au, and T.-M. Choi, "A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 5, pp. 1321–1331, 2007.
- [35] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.
- [36] Z. Deng, K.-S. Choi, L. Cao, and S. Wang, "T2fela: type-2 fuzzy extreme learning algorithm for fast training of interval type-2 TSK fuzzy logic system," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 664–676, 2014.
- [37] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice-Hall, 2001.
- [38] S. O. Olatunji, A. Selamat, and A. Abdulraheem, "A hybrid model through the fusion of type-2 fuzzy logic systems and extreme learning machines for modelling permeability prediction," *Information Fusion*, vol. 16, no. 1, pp. 29–45, 2014.
- [39] J. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," Microsoft Research Technical Report MSR-TR-98-14, 1998.
- [40] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, "Incremental support vector learning: analysis, implementation and applications," *Journal of Machine Learning Research*, vol. 7, pp. 1909–1936, 2006.
- [41] D. M. J. Tax and K.-R. Müller, "A consistency-based model selection for one-class classification," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, pp. 363–366, IEEE Computer Society, Los Alamitos, Calif, USA, August 2004.
- [42] A. G. Arkedev and E. M. Braverman, *Computers and Pattern Recognition*, Thompson, Washington, DC, USA, 1966.
- [43] L. Manevitz and M. Yousef, "One-class document classification via Neural Networks," *Neurocomputing*, vol. 70, no. 7–9, pp. 1466–1481, 2007.
- [44] M. F. Jiang, S. S. Tseng, and C. M. Su, "Two-phase clustering process for outliers detection," *Pattern Recognition Letters*, vol. 22, no. 6-7, pp. 691–700, 2001.
- [45] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, 1985.
- [46] D. M. J. Tax and R. P. W. Duin, "Data descriptions in subspaces," in *Proceedings of the International Conference on Pattern Recognition*, vol. 2, pp. 672–675, 2000.
- [47] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The VLDB Journal*, vol. 8, no. 3-4, pp. 237–253, 2000.
- [48] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Walton Street, Oxford, UK, 1995.
- [49] C.-C. Chang and C.-J. Lin, "LIBSVM: a Library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.
- [50] D. M. J. Tax, *DDtools, the Data Description Toolbox for Matlab*, 2014, [http://prlab.tudelft.nl/david-tax/dd\\_tools.html](http://prlab.tudelft.nl/david-tax/dd_tools.html).
- [51] K. Bache and M. Lichman, *UCI Machine Learning Repository*, School of Information and Computer Sciences, University of California, Irvine, Calif, USA, 2013, <http://archive.ics.uci.edu/ml.html>.
- [52] C. J. van Rijsbergen, *Information Retrieval*, Butterworths, London, UK, 1979.

## Research Article

# Anomaly Detection via Midlevel Visual Attributes

Tan Xiao,<sup>1,2</sup> Chao Zhang,<sup>1</sup> and Hongbin Zha<sup>1</sup>

<sup>1</sup>Key Laboratory of Machine Perception, Peking University, Beijing 100084, China

<sup>2</sup>CRSC Communication & Information Corporation, Beijing 100070, China

Correspondence should be addressed to Tan Xiao; [pkuxiaotan@pku.edu.cn](mailto:pkuxiaotan@pku.edu.cn)

Received 23 August 2014; Revised 18 November 2014; Accepted 24 November 2014

Academic Editor: Yi Jin

Copyright © 2015 Tan Xiao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Automatically discovering anomalous events and objects from surveillance videos plays an important role in real-world application and has attracted considerable attention in computer vision community. However it is still a challenging issue. In this paper, a novel approach for automatic anomaly detection is proposed. Our approach is highly efficient; thus it can perform real-time detection. Furthermore, it can also handle multiscale detection and can cope with spatial and temporal anomalies. Specifically, local features capturing both appearance and motion characteristics of videos are extracted from spatiotemporal video volume (STV). To bridge the large semantic gap between low-level visual feature and high-level event, we use the middle-level visual attributes as the intermediary. And these three-level framework is modeled as an extreme learning machine (ELM). We propose to use the spatiotemporal pyramid (STP) to capture the spatial and temporal continuity of an anomalous even, enabling our approach to cope with multiscale and complicated events. Furthermore, we propose a method to efficiently update the ELM; thus our approach is self-adaptive to background change which often occurs in real-world application. Experiments on several datasets are carried out and the superior performance of our approach compared to the state-of-the-art approaches verifies its effectiveness.

## 1. Introduction

*1.1. Motivation.* Surveillance systems have been widely used in the city, and detecting anomalous events from the system plays an important role in real world. However, current systems are actually quite burdensome for human operators because they are required to watch a large quantity of screens (usually up to 50 [1]) which show the content captured by several different cameras. Detecting unusual individuals and events [2], a.k.a., anomalies, is one of the most important and main tasks of human operators. Thus the performance of anomaly detection is highly dependent on the human operators. However, the quantity of cameras is growing explosively in the city requiring more and more human operators, and the task is becoming more difficult and tiring for human operators such that the performance of operators can degrade significantly [3]. Fortunately, with the development of video analytics techniques, automatic anomaly detection approaches, which can analyze video streams automatically to warn, possibly in real time, the human operators that

an anomalous event is currently taking place, have attracted considerable attention in recent years.

Specifically, anomaly detection is defined as discovering events with a low probability of occurrence from surveillance videos in computer vision community. Several approaches have been proposed in recent years and generally they can be summarized in the following three categories based on how their models are constructed, that is, supervised approaches [4–10], semisupervised approaches [11], and unsupervised approaches [12–21]. In real-world scenario, anomalies are usually quite rare, as its definition. And they show significant difference between each other and they also have unpredictable variations. Thus unsupervised approaches are more favored in most recent years. Moreover, since anomalous events are indeed difficult, almost impossible to be defined in advance, unsupervised approaches are actually practical in real-world applications.

Furthermore, unsupervised approaches can also be divided into several subcategories according to the techniques they use. Trajectories based approaches [17, 22–24]

focus on the spatial location of objects or persons and their motions are tracked. But these approaches have a significant limitation that they can only capture the abnormal track because they only consider the spatial deviations; thus an abnormal target with abnormal appearance and motion but following a normal track cannot be detected. In addition, because precise segmentation of a target is almost impossible in a crowd scene, these approaches cannot be applied to such scenario. To model typical motion of objects, optical flow has been widely utilized also [15, 16]. But as mentioned in [22], these approaches have very unstable performance in crowded scenes. And they can only detect anomalous motion while ignoring anomalous appearance because they just focus on the motion of objects and persons. Recently, densely sampled local spatiotemporal descriptor which represents both motion and appearance characteristics are utilized in [20, 21], and they can also possess some degree of robustness to unimportant variations in surveillance video. They construct models to capture the relationship between low-level visual features and high-level semantic event. Though promising results are achieved, they ignore the significant semantic gap between low-level visual features and high-level events; thus their performance is still unsatisfactory for real-world application.

Summarized from previous works, an effective and applicable anomaly detection approach should satisfy the following properties. (1) It should be definitely unsupervised because defining all anomalous events in advance is almost impossible and too burdensome for human operators to do so. (2) Both spatial and temporal anomalies, that is, both anomalies of motion and appearance, can be detected by the approach. (3) It can detect multiscale anomalies because it is also hard to know a priori the range of an anomaly, for example, its size, speed, and duration. (4) It can be online updated to self-adapt to scene change in both motion and appearance. Actually, the appearance of background is always changing in surveillance videos because of lighting condition, weather, and so forth. And the change of motion pattern should not be ignored too. (5) Last but the most important, it is able to detect the anomalies from the surveillance videos effectively and efficiently.

*1.2. Contribution.* To address this challenging problem, in this paper we propose a novel automatic anomaly detection approach with extreme learning machine (ELM) [25] based visual attribute and spatiotemporal pyramid (STP). The former one focuses on the relationship between low-level visual features and high-level event and the latter can capture the spatial and temporal continuity of an event. We propose to combine them because both parts are important for anomaly detection.

Specifically, spatiotemporal video volumes (STV) with pixel-by-pixel analysis which are densely sampled from surveillance video lay the foundation of our approach. Then spatiotemporal feature descriptor, which can capture both motion and appearance characteristics, is extracted for each STV, and each STV is further represented by bag-of-words HOG feature. Then to bridge the semantic gap between low-level visual features and high-level event, we propose to

use visual attributes as the intermediary, which is motivated by the extensive research on attribute learning for visual analysis recently [26–28]. We propose to model this three-level (feature-attribute-event) framework by extreme learning machine (ELM). The output of the ELM can be utilized to tell whether the STV belongs to an anomaly. And since the ELM can be constructed and updated with extremely high efficiency, the model can be updated continuously with coming surveillance videos; thus no offline or supervised pretraining is required for our model. So our approach can detect anomaly which even has not been observed before. And the efficient update procedure also enables our approach to cope with the scene change in both motion and appearance. Furthermore, to detect multiscale and complicated anomalies, we use spatiotemporal pyramid (STP), which is the temporal extension of spatial pyramid [29]. Thus, with different scales, multiscale event can be effectively detected by STP. Moreover, since an event is always related to several STVs which may have different location or time or both, complicated events can be detected by discovering the relationship. STP can achieve this task, enabling our approach to detect complicated events. Thus the proposed approach can effectively and accurately detect multiscale and complicated anomalies, both in motion and appearance. And the model can be quite efficiently constructed and updated in an unsupervised way; thus it can detect anomalies which have not been observed before. Moreover, our approach is self-adaptive to background change in both motion and appearance.

We show the framework overview of the proposed approach in Figure 1. The input is a video stream. Then the stream is densely sampled; that is, it is sampled pixel by pixel. Around each pixel, a 3D volume is constructed, that is, STV. This volume is segmented with no overlap into 8 smaller STVs. Thus the upper and coarser level of STP is formed by the larger STV which can capture the overall information of an event. And the smaller STVs form the lower but finer level of STP, which can capture the details of an event. Actually, we can continue segmenting any small STV into another 8 smaller STVs. But this leads to much more computational complexity and we find that a two-level STP can achieve satisfactory performance for anomaly detection. Then, for each STV, HOG features which can capture both motion and appearance characteristics of STV are extracted. The HOG of upper level STV can be efficiently constructed from its lower level STVs; thus our multilevel STP actually does not require too much extra computation for feature extraction, which is an essential property for real-time detection. Instead of learning a model that directly connect low-level visual features to high-level event which may suffer from the large semantic gap between them, we propose to use visual attribute [26] as the middle level to bridge the semantic gap. And we utilize extreme learning machine to model this three-level framework; that is, low-level feature is the input, visual attribute is the hidden unit, and the output of ELM can be regarded as the high-level event. And ELM can be efficiently trained and updated which is also important. Finally, the anomaly judgement is given based on the results of both levels.

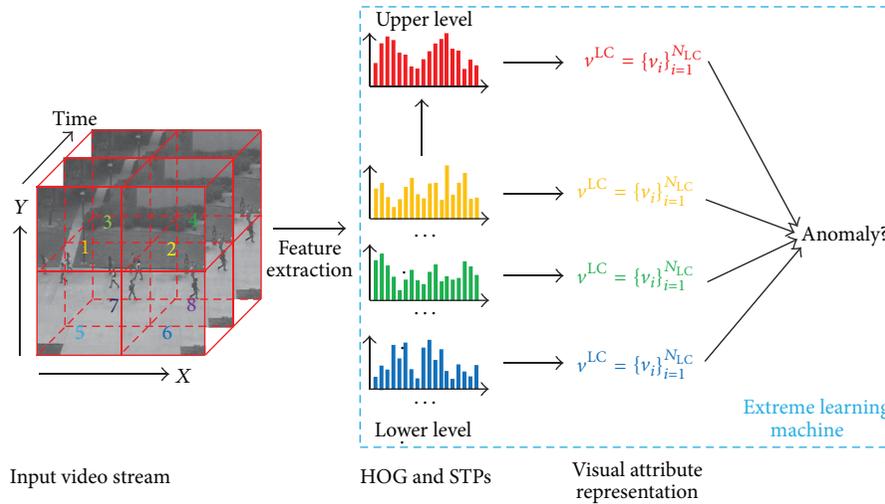


FIGURE 1: Overview of our approach. This is an example of two-level spatiotemporal pyramid. The input is a video stream. Then a 3D volume around a pixel is constructed represented by the outer red cube. Then it is segmented into 8 ( $2 \times 2 \times 2$ ) smaller cubes denoted by different numbers in this figure. The smaller cubes form the lower but finer level of the pyramid. HOG features are extracted for each smaller cube. And the HOG features of upper level cube can be constructed efficiently from lower level cubes. We use visual attribute representation to bridge the semantic gap between low-level feature and high-level event. The three-level (feature-attribute-event) framework can be modeled by extreme learning machine. Finally the anomaly detection is completed by combining the outputs of the machine.

In a summary, we make the following contributions to this paper.

- (i) A novel approach for automatic anomaly detection is proposed. It is based on densely sampled STVs. Visual attribute is utilized to bridge the semantic gap and we use ELM to model this three-level framework. ELM can also effectively and efficiently tell whether a STV belongs to an anomalous event.
- (ii) We propose to use spatiotemporal pyramid (STP) to capture the spatial and temporal continuity of an anomalous event. And our approach can perform multiscale detection with the presence of STP.
- (iii) The use of ELM has another important benefit; that is, it can be efficiently updated. Thus our approach can perform efficient online update and thus can adaptively learn the event patterns in the scene and cope with the scene change of both motion and appearance.
- (iv) Extensive experiments on several public datasets are carried out to evaluate the effectiveness of our approach for anomaly detection. And the results show that our approach can achieve satisfactory performance and significantly outperform several state-of-the-art approaches.

The rest of this paper is organized as follows. In Section 2, we will briefly review some works related to our approach. The details for detection via ELM-based visual attribute and online updating method are described in Section 3. We will introduce the spatiotemporal pyramid in Section 4. The experiments and results are presented in Section 5 and we draw conclusions in Section 6.

## 2. Related Work

As has been mentioned in Section 1, one of the most widely used techniques in previous works is trajectory analysis. But generally, precise tracking methods [30, 31] are always required by this kind of methods. But unfortunately, tracking objects is quite time-consuming and computationally expensive, especially in crowded scenes where a large number of objects and persons are moving such that precise segmentation of targets is nearly impossible, which is the foundation of tracking analysis. And some previous works also utilize optical flow [15, 16], but their performance in crowded scenes is quite unreliable [20].

Recently, the focus of anomaly detection has turned from object detection or tracking to local spatiotemporal features. Several approaches have been proposed and received increasing attention [32, 33]. Typically, these approaches focus on pixel level. They propose to describe the local characteristic of video by low-level visual features, like color, texture, and motion. Then they can construct pixel-level background model and behavior template based on the local features [34–37]. In addition, approaches utilizing spatiotemporal video volumes in the context of bag-of-video-words have achieved promising results [13, 20, 38]. For example, probabilistic models such as latent Dirichlet allocation (LDA) [39] can be straightforwardly applied to video analysis if we ignore the spatial and temporal relationship of local features [40, 41]. But intuitively, the spatial and temporal relationship between STVs are quite essential for scene understanding and event detection [42]. Noticing this point, the efforts to incorporate either spatial or temporal composition of STVs into the conventional probabilistic model have been made in some works. But they are not able to handle online and real-time detection because they are highly time-consuming

and computationally expensive [40]. In addition, several approaches [32, 35, 43, 44] try to construct models based on the spatiotemporal behavior and analyse the spatiotemporal pattern of each pixel as a function of time to detect low-level local anomalous events. However, they ignore the relationship between each pixel in space and time because they just process each pixel independently such as in [43], which may lead to too local detection.

A multiscale and nonparametric approach is proposed in [20] to perform real-time anomaly detection and localization. Dense and local spatiotemporal features which can capture both motion and appearance characteristics of objects are extracted at each individual pixel. And to take advantage of the spatial and temporal relationship between pixels, ‘‘overlapping’’ features are utilized in their approach. As reported in their paper, they can achieve promising results, but their approach indeed faces the challenge of efficiency when performing accurate multiscale anomaly detection. Actually, our spatiotemporal pyramid is partially motivated by their overlapping features because we both consider the spatial and temporal relationship between features. But compared to their overlapping features, our STP can be constructed with much more efficiency and much better performance can be achieved. Furthermore, our STP can cope with multiscale detection naturally while their approach actually treats different scales independently.

Moreover, approaches mentioned above all construct models between low-level visual features and high-level events straightforwardly while ignoring the semantic gap between them which is quite important for event detection. Thus they cannot achieve satisfactory results. This problem motivates us to use visual attribute as intermediary to bridge the semantic gap for more accurate detection.

### 3. Detection via ELM-Based Visual Attribute

**3.1. Spatiotemporal Local Features.** Tracking objects or persons in videos is quite time consuming and its performance is unstable under crowded scene. Instead, we utilize the local features which can capture the local spatial and temporal characteristics of video. By analyzing the pattern of spatiotemporal local features, we can detect and localize the anomalies.

First of all, we need to extract meaningful local features to capture the motion and appearance characteristics of densely sampled STVs at each pixel. Considering a pixel  $(x, y, t)$ , we can construct a STV  $v \in \mathbb{R}^{n_x \times n_y \times n_t}$  with the size  $n_x \times n_y \times n_t$  centered at  $(x, y, t)$ , where  $n_x \times n_y$  denotes the size of spatial window and  $n_t$  is the depth of STV in time. Typically,  $5 \times 5 \times 5$  or  $10 \times 10 \times 10$  can be proper size of a STV. Then we calculate the histogram of the spatiotemporal gradient (HOG) of the video in polar coordinates to describe the STV, following the works in [20, 21, 45]. Now we can denote the spatial gradients as  $G_x(x, y, t)$ ,  $G_y(x, y, t)$ , and the temporal gradient as  $G_t(x, y, t)$ , respectively, at pixel  $(x, y, t)$ . Specifically, they are computed using the finite difference approximations as follows:

$$G_x(x, y, t) = L_{\sigma_d}(x + 1, y, t) - L_{\sigma_d}(x - 1, y, t),$$

$$G_y(x, y, t) = L_{\sigma_d}(x, y + 1, t) - L_{\sigma_d}(x, y - 1, t),$$

$$G_t(x, y, t) = L_{\sigma_d}(x, y, t + 1) - L_{\sigma_d}(x, y, t - 1), \quad (1)$$

where  $L_{\sigma_d}$  is obtained by filtering the signal with a Gaussian kernel of bandwidth  $\sigma_d$  to suppress the noise. In our experiment, we find out that setting  $\sigma_d = 1.1$  leads to satisfactory performance.

In real-world applications, surveillance videos are always affected by noise and the effect of local texture and contrast also have significant influence on the video analysis. Thus, in order to alleviate the influence of noise in videos and texture and contrast, we first normalize the spatial gradient as follows:

$$G_s(x, y, t) = \frac{\sqrt{G_x^2(x, y, t) + G_y^2(x, y, t)}}{\sum_{x', y', t' \in v} \sqrt{G_x^2(x', y', t') + G_y^2(x', y', t') + \epsilon}}, \quad (2)$$

where  $G_s(x, y, t)$  is the normalized spatial gradient and  $\epsilon$  is a small constant to avoid numeric instabilities (denominator is equal to zero by chance). Typically we can set  $\epsilon = 0.01$ . Based on the normalized spatial gradient, we can further construct 3D normalized gradient represented in polar coordinates as follows:

$$M_{3D}(x, y, t) = \sqrt{G_s^2(x, y, t) + G_t^2(x, y, t)},$$

$$\theta(x, y, t) = \tan^{-1} \left( \frac{G_y(x, y, t)}{G_x(x, y, t)} \right), \quad (3)$$

$$\phi(x, y, t) = \tan^{-1} \left( \frac{G_t(x, y, t)}{G_s(x, y, t)} \right),$$

where  $M_{3D}(x, y, t)$  is the magnitude of 3D normalized gradient and  $\phi(x, y, t) \in [-\pi/2, \pi/2]$  and  $\theta(x, y, t) \in [-\pi, \pi]$  are the orientations of the gradient, respectively. Then we can construct the histogram of oriented gradients (HOG features) for a given STV  $v$  in the following way. First, for each pixel in the give STV  $v$ , we can extract 3D normalized gradient features for them. Then the feature for each pixel is quantized into  $n_\phi + n_\theta$  bins based on their gradient orientations. Typically we can set  $n_\phi = 8$  and  $n_\theta = 16$ . So the HOG features for STV  $v$ , denoted by  $h$ , has 24 dimensions under this setting. If we look back to the feature extraction and construction procedure, we can observe that the local characteristics of both motion and appearance in the video can be captured by the HOG features. Consequently, both anomalous actions and objects can be detected based on the HOG features. Moreover, because of the normalization step at first, it shows robustness to data noise and unimportant variations in the video such as texture and contrast. Actually, this HOG feature can be used as the input of the extreme learning machine.

Besides the effectiveness of low-level visual features, we also care about the efficiency of feature extraction and construction. Though it seems that the feature construction is

quite computationally expensive, actually we can notice that it is quite efficient because the computation can be always reused as discussed below.

For example, the 3D normalized gradient at one pixel will be used by  $n_x \times n_y \times n_t$  times because it is used to construct HOG features for any STV containing it. If the 3D normalized gradient is repeatedly computed for all STVs, it is indeed burdensome. But we can precompute it in advance and we just need to reuse the obtained result for each STV. Also, the histogram of pixel  $(x, y, t)$ , denoted by  $h(x, y, t)$ , can also be precomputed by quantization a priori. Then the histogram of a STV  $v$  around pixel  $(x, y, t)$  can be computed by simply summing up all the histograms in this STV as

$$h_v(x, y, t) = \sum_{(x', y', t') \in v} h(x', y', t'). \quad (4)$$

It is obvious that just summing up all the histograms can be quite efficient. On the other hand, if we compute the HOG feature for each STV independently, it can be highly computational too. But we can observe that we can use its neighbor's HOG feature to construct the HOG feature for a given STV as follows; thus computations can be saved markedly:

$$\begin{aligned} h_{v_2} &= h_{v_1} - \sum_{(x', y', t') \in v_1 \setminus v_2} h(x', y', t') \\ &+ \sum_{(x', y', t') \in v_2 \setminus v_1} h(x', y', t'), \end{aligned} \quad (5)$$

where “ $\setminus$ ” is the set minus operation and the STVs around  $(x, y, t)$  and  $(x+1, y, t)$  are denoted by  $v_1$  and  $v_2$ , respectively. It is clear that  $v_1 \setminus v_2$  is much smaller than  $v_1$ . Moreover, the HOG feature of upper level STV can be computed by summing up the HOG features of lower level STVs in the same STP. Consider the outer red cube in Figure 1. Given the HOG of eight lower level STVs in this cube, the HOG of the upper level, that is, the red cube, can be computed as follows:

$$\begin{aligned} h_{v_{\text{up}}} &= \sum_{(x', y', t') \in v_{\text{up}}} h(x', y', t') \\ &= \sum_{i=1}^8 \sum_{(x', y', t') \in v_i} h(x', y', t') = \sum_{i=1}^8 h_{v_i}. \end{aligned} \quad (6)$$

Consequently, because of the computational tricks above, the extraction and construction of the spatiotemporal feature can be highly efficient, which is one of the most important requirements for real-time detection.

**3.2. Constructing Normal Events.** As mentioned in Section 1, our approach is unsupervised. Thus we need to define normal and anomalous events automatically. As the definition of anomaly, it is quite rare compared to normal events. Thus, the number of STVs associated with anomalies is much fewer than the number of STVs belonging to normal events in video. Generally, STVs belonging to normal events may form clusters in the feature space while STVs of anomalous events

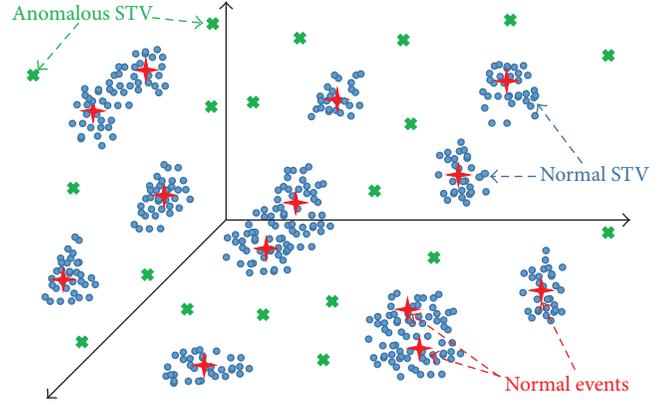


FIGURE 2: Constructing normal events. Generally, STVs belonging to normal events may form clusters in the feature space while STVs of anomalous events are outliers in the space. We can use this property to construct normal events.

are outliers in the space. Thus we can construct normal events by using this property which is illustrated in Figure 2. Consequently, we can regard the clusters as the normal events and the judgement of anomaly can be given by analyzing how close a STV is related to a cluster. Intuitively, we can construct these clusters by some clustering methods such as k-means. However, there are some parameters for clustering algorithm; for example, we need to specify  $k$  for k-means clustering. And we find out that our approach is a little sensitive to this parameter.

Instead, we propose to construct normal events automatically from video data. Given a set of spatiotemporal features  $\mathbf{H} = [h_1, \dots, h_n] \in \mathbb{R}^{d \times n}$ , where  $d = 24$  is the dimension of feature and  $n$  is the size of feature set. Actually we do not need a training set because the initial feature set  $\mathbf{H}$  can be constructed by using the first few seconds of the video. In addition, to guarantee that our selection algorithm is computationally feasible, we can also randomly select some features to reduce  $n$ . In real-world scenario,  $n$  can be tuned from 10,000 to 20,000 according to the resolution of videos. Then we need to select some features from  $\mathbf{H}$  as the representatives of clusters, that is, the normal events. In our method, the number of clusters is determined automatically by the algorithm, which is self-adaptive to the test data. Following the idea in [19], we would like to select an optimal subset of  $\mathbf{H}$  as the clusters, such that we can well reconstruct the rest of features from them. This criterion can be formulated as follows:

$$\min_{\mathbf{S}} = \frac{1}{2} \|\mathbf{H} - \mathbf{H}\mathbf{S}\|_F^2 + \lambda \|\mathbf{S}\|_{2,1}, \quad (7)$$

where  $\mathbf{S} \in \mathbb{R}^{n \times n}$  is the selection matrix,  $\|\mathbf{S}\|_F = \sqrt{\sum_i \sum_j \mathbf{S}_{ij}^2}$  is the Frobenius norm of matrix  $\mathbf{S}$ ,  $\|\mathbf{S}\|_{2,1} = \sum_{i=1}^n \|\mathbf{S}_i\|_2$  is the  $L_{2,1}$ -norm of matrix, and  $\lambda$  is the model parameter. Finally, the selection can be done by selecting index  $i$  which satisfies  $\|\mathbf{S}_i\| > 0$ . Consequently we can obtain clusters representing the normal events.

To solve this problem, we follow the method proposed in [46]. Consider an objective function  $f_0(x) = f(x) + g(x)$  where  $f(x)$  is convex and smooth and  $g(x)$  is convex but nonsmooth. The key step is to construct  $p_{Z,L}(x) = f(Z) + \langle \nabla f(Z), x - Z \rangle + (L/2)\|x - Z\|_F^2 + g(Z)$  to approximate  $f_0(x)$  at point  $Z$ . Obviously, we can define  $f(\mathbf{S}) = (1/2)\|\mathbf{H} - \mathbf{HS}\|_F^2$  and  $g(\mathbf{S}) = \lambda\|\mathbf{S}\|_{2,1}$ . So we can construct  $p_{Z,L}(\mathbf{S})$  as

$$p_{Z,L}(\mathbf{S}) = f(\mathbf{Z}) + \langle \nabla f(\mathbf{Z}), \mathbf{S} - \mathbf{Z} \rangle + \frac{L}{2} \|\mathbf{S} - \mathbf{Z}\|_F^2 + g(\mathbf{Z}). \quad (8)$$

And we can define another function  $D_\tau(\cdot) : \mathbf{M} \in \mathbb{R}^{n \times n} \mapsto \mathbf{N} \in \mathbb{R}^{n \times n}$ :

$$\mathbf{N}_i = \begin{cases} 0, & \|\mathbf{M}_i\| \leq \tau \\ \left(1 - \frac{\tau}{\|\mathbf{M}_i\|}\right) \mathbf{M}_i, & \text{otherwise.} \end{cases} \quad (9)$$

Theoretically,  $\mathbf{S}$  is given by solving the following optimization problem:

$$\mathbf{S} = \arg \min_{\mathbf{S}} p_{Z,L}(\mathbf{S}). \quad (10)$$

And this optimization problem can be equivalently written as follows:

$$\begin{aligned} & \min_{\mathbf{S}} f(\mathbf{Z}) + \langle \nabla f(\mathbf{Z}), \mathbf{S} - \mathbf{Z} \rangle + \frac{L}{2} \|\mathbf{S} - \mathbf{Z}\|_F^2 + \lambda \|\mathbf{S}\|_{2,1} \\ & \iff \min_{\mathbf{S}} \frac{L}{2} \left\| \left( \mathbf{S} - \mathbf{Z} + \frac{1}{L} \nabla f(\mathbf{Z}) \right) \right\|_F^2 + \lambda \|\mathbf{S}\|_{2,1} \\ & \iff \min_{\mathbf{S}} \frac{L}{2} \left\| \mathbf{S} - \left( \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) \right) \right\|_F^2 + \lambda \|\mathbf{S}\|_{2,1} \\ & \iff \min_{\mathbf{S}} \frac{L}{2} \left\| \mathbf{S} - \left( \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) \right) \right\|_F^2 + \lambda \sum_{i=1}^n \|\mathbf{S}_i\|_2. \end{aligned} \quad (11)$$

And since the  $L_2$  norm is self-dual, the problem can be further rewritten as follows by introducing a dual variable  $\mathbf{Y} \in \mathbb{R}^{n \times n}$ :

$$\begin{aligned} & \min_{\mathbf{S}} \frac{L}{2} \left\| \mathbf{S} - \left( \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) \right) \right\|_F^2 + \lambda \sum_{i=1}^n \max_{\|\mathbf{Y}_i\| \leq 1} \langle \mathbf{Y}_i, \mathbf{S}_i \rangle \\ & \iff \max_{\|\mathbf{Y}_i\| \leq 1} \min_{\mathbf{S}} \frac{L}{2} \left\| \mathbf{S} - \left( \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) \right) \right\|_F^2 + \lambda \sum_{i=1}^n \langle \mathbf{Y}_i, \mathbf{S} \rangle \\ & \iff \max_{\|\mathbf{Y}_i\| \leq 1} \min_{\mathbf{S}} \frac{L}{2} \left\| \mathbf{S} - \left( \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) - \frac{\lambda}{L} \mathbf{Y} \right) \right\|_F^2 \\ & \quad - \frac{1}{2} \left\| \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) - \frac{\lambda}{L} \mathbf{Y} \right\|_F^2. \end{aligned} \quad (12)$$

The second equation above is obtained by swapping ‘‘min’’ and ‘‘max.’’ Because this function is convex with respect to  $\mathbf{S}$  and concave with respect to  $\mathbf{Y}$ , this swapping will not change the problem by Von Neumann minimax theorem. Further,

denote  $\mathbf{S} = \mathbf{Z} - (1/L)\nabla f(\mathbf{Z}) - (\lambda/L)\mathbf{Y}$ . From the last equation above we can obtain an equivalent problem as follows:

$$\max_{\|\mathbf{Y}_i\| \leq 1} -\frac{1}{2} \left\| \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) - \frac{\lambda}{L} \mathbf{Y} \right\|_F^2. \quad (13)$$

Analogous to the substitution above, let  $\mathbf{Y} = -(L/\lambda)(\mathbf{S} - \mathbf{Z} + (1/L)\nabla f(\mathbf{Z}))$ ; we can change the problem above into a problem in terms of the original variable  $\mathbf{S}$  as

$$\begin{aligned} & \min_{\|(L/\lambda)(\mathbf{S} - \mathbf{Z} + (1/L)\nabla f(\mathbf{Z}))\|_F \leq 1} \|\mathbf{S}\|_F^2 \\ & \iff \sum_{i=1}^n \min_{\|\mathbf{S}_i - (\mathbf{Z} - (1/L)\nabla f(\mathbf{Z}))\|_2 \leq \lambda/L} \|\mathbf{S}_i\|_2^2. \end{aligned} \quad (14)$$

Therefore, the optimal solution to the first problem above is equivalent to the last problem above. Actually, we can optimize each row in  $\mathbf{S}$  independently in the last problem. Considering each row of  $\mathbf{S}$ , respectively, we can get the closed form as

$$\mathbf{S} = \arg \min_{\mathbf{S}} p_{Z,L}(\mathbf{S}) = D_{\lambda/L} \left( \mathbf{Z} - \frac{1}{L} \nabla f(\mathbf{Z}) \right). \quad (15)$$

We summarize the whole algorithm in Algorithm 1.

**3.3. Detection via ELM-Based Visual Attributes.** In the above two subsections, we introduce how to extract low-level visual features which can simultaneously capture both motion and appearance characteristics of videos and how to construct a set of normal events in an unsupervised way. Intuitively, one simple way to tell whether a STV represented by low-level visual feature belongs to a anomaly is to consider the relationship between the low-level feature and all high-level normal events. In fact, if it is closely related to one event, it has very high probability to be normal. On the contrary, if it does not have strong relation to any one normal events, it usually belongs to an anomaly. Thus we just need to define a function or model to measure the relationship between low-level visual features and high-level events. However, directly constructing model between low-level visual features and high-level events may suffer from the large semantic gap between them. Recent research on visual attribute [26–28] has pointed out this problem and proposed to use visual attribute as the intermediary to bridge the semantic gap.

The basic idea is shown in Figure 3. Instead of constructing model between low-level visual features and high-level events directly, now we need construct two models, that is, one between low-level visual features and middle-level visual attributes and one between middle-level visual attributes and high-level events. Actually, directly connecting low-level visual features and high-level events in only one step is quite difficult. Thus we use two steps such that either one can be more feasible.

Theoretically, we can apply any models to both steps. And we find that linear model can always lead to satisfactory performance while guaranteeing the efficiency for both training and detection given a set of STVs and the corresponding low-level visual features  $\mathbf{H} = [h_1, \dots, h_n] \in \mathbb{R}^{d \times n}$ , and the

**Input:**  $\mathbf{H}, \lambda = 1, \mathbf{S}_0, K, c$   
**Output:**  $\mathbf{S}$   
(1) Initialize  $\mathbf{Z}_0 = \mathbf{S}_0, a_0 = 1$   
(2) **for**  $k = 0, 1, 2, \dots, K$  **do**  
(3)  $\mathbf{S}_{k+1} = \arg \min_{\mathbf{S}} : p_{\mathbf{Z}_k, L}(\mathbf{S}) = D_{\lambda/L} \left( \mathbf{Z}_k - \frac{1}{L} \nabla f(\mathbf{Z}_k) \right)$   
(4) **while**  $f_0(\mathbf{S}_{k+1}) > p_{\mathbf{Z}_k, L}(\mathbf{S}_{k+1})$  **do**  
(5)  $L = \frac{L}{c}$   
(6)  $\mathbf{S}_{k+1} = \arg \min_{\mathbf{S}} : p_{\mathbf{Z}_k, L}(\mathbf{S}) = D_{\lambda/L} \left( \mathbf{Z}_k - \frac{1}{L} \nabla f(\mathbf{Z}_k) \right)$   
(7) **end while**  
(8)  $a_{k+1} = \frac{(1 + \sqrt{1 + 4a_k^2})}{2}$   
(9)  $\mathbf{Z}_{k+1} = \left( \frac{a_{k+1} + a_k - 1}{a_{k+1}} \right) \mathbf{S}_{k+1} - \left( \frac{a_k - 1}{a_{k+1}} \right) \mathbf{S}_k$   
(10) **end for**

ALGORITHM 1: Normal events construction.

corresponding event labels  $\mathbf{E} = [e_1, \dots, e_n] \in \{-1, 1\}^{m \times n}$ , where  $e_{ij} = 1$  if the  $j$ th STVs belongs to the  $i$ th event and  $e_{ij} = -1$  otherwise. Previous approaches directly construct models between  $\mathbf{H}$  and  $\mathbf{E}$  which may suffer from the semantic gap between them. Here we propose to utilize a middle-level  $\mathbf{A} = [a_1, \dots, a_n] \in \mathbb{R}^{k \times n}$  to bridge the semantic gap, where  $k$  is the number of visual attributes which we set as  $k = 256$  in this paper. Actually,  $a_j$  can be regarded as the visual attribute for the  $i$ th STV. Thus we can construct two linear models as follows:

$$a_{ij} = g(w_i \cdot h_j + b_i), \quad i = 1, \dots, k \quad (16)$$

$$e_{ij} = \sum_{t=1}^k \beta_{it} \cdot a_{tj}, \quad i = 1, \dots, m,$$

where  $a_{ij}$  is the  $i$ th attribute for the  $j$ th STV and  $e_{ij}$  is the relation degree between the  $i$ th event and the  $j$ th STV which is used to show that the STV belongs to an anomaly or not. Now we just need to specify two model parameters, that is,  $\mathbf{W} = [w_1, \dots, w_k] \in \mathbb{R}^{d \times k}$  and  $\mathbf{B} = [b_1, \dots, b_k]^T \in \mathbb{R}^k$  for the model between low-level visual feature and middle-level visual attribute and  $\beta = [\beta_1, \dots, \beta_k] \in \mathbb{R}^{m \times k}$  for the model between middle-level visual attribute and high-level event. And  $g(\cdot)$  is the activation function which is infinitely differentiable, such as sigmoidal function, radial basis, cosine, and exponential. In this paper we use sigmoidal function as the activation function.

Now we need to minimize the following objective function to determine model parameters:

$$O = \sum_{j=1}^n \left\| \sum_{i=1}^k \beta_i \cdot g(w_i \cdot h_j + b_i) - e_j \right\|_F^2, \quad (17)$$

where  $n$  is the number of training data. As mentioned before, our approach actually does not need prepared training data. We can use the normal event set constructed above as the training data. Suppose finally we select  $n$  events with

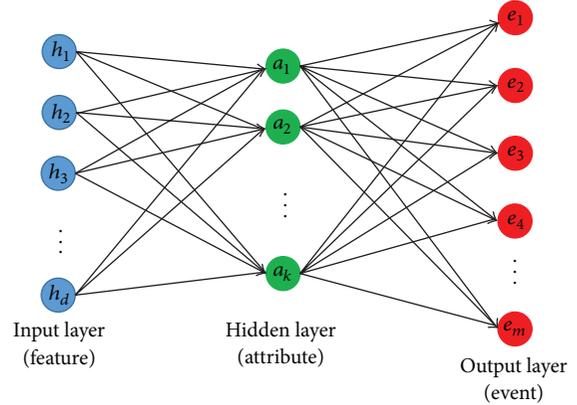


FIGURE 3: ELM-based visual attribute. There is large semantic gap between low-level visual feature and high-level event. To bridge the gap, we can use visual attribute as intermediary. And this three-level (feature-attribute-event) framework can be formulated as an extreme learning machine.

corresponding feature representation. Then the features can be used as  $\mathbf{H}$ , and each event has its own representation  $e_j$ , where  $e_{jj} = 1$  and  $e_{ij} = -1$  for  $i \neq j$ . Thus we can construct  $\mathbf{H}$  and  $\mathbf{E}$  as the training set in an unsupervised way while requiring no extra effort.

In fact, learning two models simultaneously is quite difficult and inefficient because we need to adjust parameters in two models iteratively, such that it cannot be applied to online and real-time applications. But fortunately, this three-level (feature-attribute-event) framework can be formulated as an extreme learning machine (ELM) [25, 47–49] which is a variant of artificial neural network (ANN), where feature is the input layer, attribute is the hidden layer, and event is the output layer. In the theory of ELM, the parameters between input layer and hidden layer can be totally random; that is, we actually do not need to learn these parameters from the training data. Thus we just need to compute the parameters between the hidden layer and the output layer,



FIGURE 4: Experiments on Bellevue dataset.

which is extremely fast compared to conventional ANN which is solved by backpropagation.

Now we can randomize the parameters  $\mathbf{W}$  and  $\mathbf{B}$  for the model between low-level visual feature and middle-level visual attribute. Because we will not change  $\mathbf{W}$  and  $\mathbf{B}$ , we can compute the visual attributes for the training data as follows:

$$\mathbf{A} = g(\mathbf{W}^T \mathbf{H} + \mathbf{B} \mathbf{1}_k^T), \quad (18)$$

where  $\mathbf{1}_k = [1, \dots, 1]^T$ . Then we just need to compute  $\beta$  by minimizing the following objective function:

$$O = \|\beta \mathbf{A} - \mathbf{E}\|_F^2 \quad (19)$$

and we can get the solution for  $\beta$  as

$$\hat{\beta} = \mathbf{E} \mathbf{A}^\dagger, \quad (20)$$

where  $\mathbf{A}^\dagger$  is the Moore-Penros generalized inverse of matrix  $\mathbf{A}$  [50, 51]. Then, given any STV represented by low-level visual features  $h^* \in \mathbb{R}^d$ , we can compute its relationship with any event as  $e^* \in \mathbb{R}^m$  via the middle-level visual attribute by the ELM as follows:

$$e^* = \hat{\beta} g(\mathbf{W}^T h + \mathbf{B}). \quad (21)$$

Now we need to discuss how to tell whether this STV belongs to an anomaly based on  $e^*$ . Actually, as discussed above, a normal STV is always close to a cluster while an abnormal STV is always outlier. And a normal event is represented by a cluster. Furthermore,  $e_i^*$  denotes the relationship between this STV with the  $i$ th event (i.e., cluster). Thus if it belongs to a normal event, it may have strong relationship with an event, leading to  $e_i^* \approx 1$  for some  $j$  implying strong relationship, while no elements in  $e^*$  may have large value because it is an outlier and has quite weak relationship with all events. Consequently, we can define the degree of anomaly as

$$d_{\text{anomaly}} = 1 - \max_i(e_i^*). \quad (22)$$

Formally we can select an anomaly threshold  $\delta$  such that a STV that satisfies  $d_{\text{anomaly}} > \delta$  is judged as abnormal while that satisfies  $d_{\text{anomaly}} \leq \delta$  is judged as normal. Now we need to determine the value of  $\delta$ . We can define the anomaly probability  $p_a$ , which is empirically selected from

$10^{-2}$ ,  $10^{-3}$ ,  $10^{-4}$ , and  $10^{-5}$ , depending on the user's need. High true positive rate and high false positive rate are achieved with a large  $p_a$  while a small  $p_a$  will lower both. Then we can compute  $d_{\text{anomaly}}$  for all STVs in the first one or two seconds in a test video and set  $\delta$  such that the ratio of STVs whose  $d_{\text{anomaly}}$  are larger than  $\delta$  is about  $p_a$ . So about  $p_a$  STVs will be treated as anomalies. We have a postprocessing step on the initial judgement to obtain better results, which will be introduced in detail in Section 4.

So far, we complete the introduction to our anomaly detection algorithm based on visual attribute and extreme learning machine.

**3.4. An Alternative.** We construct our model based on ELM to model this three-level frame work as above. And we propose to randomize  $\mathbf{W}$  and choose an infinitely differentiable activation function. The infinitely differentiable activation function always leads to continuous output, that is, real-value attribute. However, recent research demonstrates that binary attributes can lead to better performance. Thus we want the activation function to output binary value, such that we can use the sign function as the activation function defined as follows:

$$g(x) = \text{sign}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{otherwise.} \end{cases} \quad (23)$$

In our experiment, we find out that this activation function shows satisfactory performance as previous works [28]. However, as the sign function is not infinitely differentiable, we cannot simply randomize  $\mathbf{W}$  as in last section. In this paper, we propose an alternative for learning  $\mathbf{W}$  which is also quite effective and efficient, such that we can use sign function as the activation function.

First we can construct the training set with features  $\mathbf{H} = [h_1, \dots, h_n]$  and the corresponding event representations  $\mathbf{E} = [e_1, \dots, e_n]$ , where  $e_{jj} = 1$  and  $e_{ij} = -1$  for  $i \neq j$ , as mentioned in the last section. Then we need to construct the binary attributes  $\mathbf{A} = [a_1, \dots, a_n] \in \{-1, 1\}^{k \times n}$  and the model parameters  $\mathbf{W}$ .

To construct effective visual attributes, we think two important principles should be followed: (1) the learned attributes should be predictable; that is, we can generate the attribute representation correctly from low-level visual features and (2) they should be discriminative; that is, different events should have different attribute representations.

**Input:**  $H, C = \gamma = 1, k$   
**Output:**  $A, W$ ;  
(1) Initialize  $W$  by randomization  
(2) Initialize  $A: a_j^i = \text{sign}(w_i' \cdot h_j), \forall j = 1, \dots, n, i = 1, \dots, k$   
(3) **repeat**  
(4) Optimize  $A$  in  $\min_A \sum_{i=1}^n \sum_{j \neq i} d(a_i, a_j)$  by block gradient descent  
(5) Train  $k$  linear SVMs to update  $w_i, \forall i = 1, \dots, k$ , using  $a_j^i$  as the label for feature  $h_j$  and the  $i$ th attribute,  $\forall j = 1, \dots, n, i = 1, \dots, k$   
(6) Update  $A: a_j^i = \text{sign}(w_i' \cdot h_j), \forall j = 1, \dots, n, i = 1, \dots, k$   
(7) **until** Convergence  
(8) Return  $A, W$

ALGORITHM 2: Learning visual attributes and model parameters.

It is difficult to construct effective model between low-level visual features and high-level events because of the semantic gap between them; thus we utilize visual attributes as the intermediary such that the models between attributes and features or events can be more effective. In fact, two principles proposed above reflect the property that the visual attributes should have as the intermediary.

More specifically, we propose to use max margin models for linear support vector machines (SVM) such that the attributes can be reliably predicted from the original low-level visual features. And we require the distance between the attribute representations of different events to be large; thus enough margin between events can be provided by the visual attributes. Thus, the objective function for learning visual attributes and the model parameters  $W$  can be written as follows:

$$\begin{aligned} \min_{W, A} \quad & \sum_{i=1}^k \|w_i\|_F^2 + C \sum_{j=1}^n \sum_{i=1}^k \xi_j^i - \frac{\gamma}{2} \sum_{i=1}^n \sum_{j \neq i} d(a_i, a_j) \\ \text{s.t.} \quad & \xi_j^i \geq 0 \quad \forall j = 1, \dots, n, i = 1, \dots, k \\ & a_j^i (w_i' \cdot h_j) \geq 1 - \xi_j^i \quad \forall j = 1, \dots, n, i = 1, \dots, k \\ & a_j^i = g(w_i' \cdot h_j) = \text{sign}(w_i' \cdot h_j) \\ & \forall j = 1, \dots, n, i = 1, \dots, k. \end{aligned} \quad (24)$$

As mentioned above, we choose sign function as the activation function such that we can obtain binary visual attributes. And because each event is related to only one feature and vice versa, we just need the distance between all attribute representations to be as large as possible, leading to large margin between events. And  $d(x, y)$  is the distance measure for two binary vectors  $x$  and  $y$ . Typically, Hamming distance, which counts the number of different bits between  $x$  and  $y$ , is utilized as the distance measure.  $w_i$  is the weight vector corresponding to the  $i$ th visual attribute,  $\xi_j^i$  is the slack variable corresponding to the  $i$ th attribute for the  $j$ th feature (event), and  $C$  and  $\gamma$  are the model parameters to balance the weight of large margin between attributes, predictability, and large margin between events.

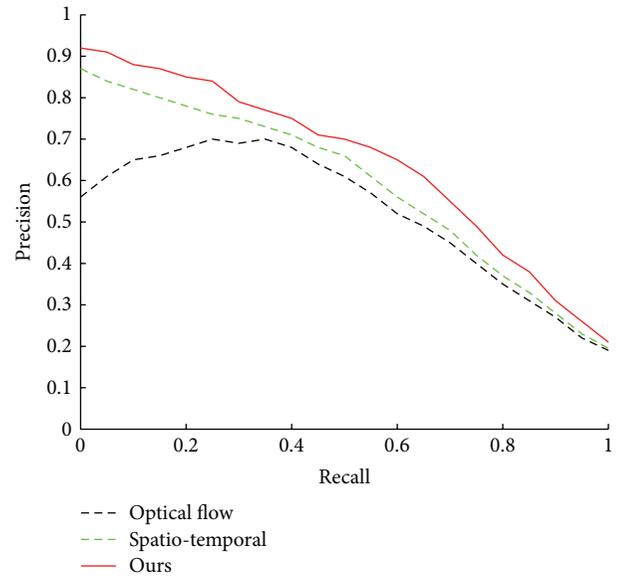


FIGURE 5: Performance curves on Bellevue dataset.

This optimization problem is quite difficult to reach global minimum. But fortunately, local minimum can result in satisfactory performance. Here we propose to adopt an iterative strategy for the optimization problem consisting of three steps. At first, we *adjust* the value of  $A$  while keeping  $W$  fixed. The purpose of this step is to improve the margin between events represented by visual attributes, that is,  $A$ . Because  $A$  is binary, we can use block gradient descent for this step. Secondly, we update the model parameters  $W$  while fixing  $A$ . Actually, when  $A$  is fixed, the optimization problem for  $W$  can be regarded as learning  $k$  independent support vector machines. At the third step, we *update*  $A$  by the input low-level visual feature  $H$  and model parameter  $W$  such that the learned model parameters can indeed generate correct prediction. Here we need to point out the difference between the first and the third steps. The first step is to construct more discriminative attributes while the third step aims to make them predictable. They are quite different, but both are important. The learning algorithm is summarized in Algorithm 2.



FIGURE 6: Experiments on Boat-Holborn dataset.

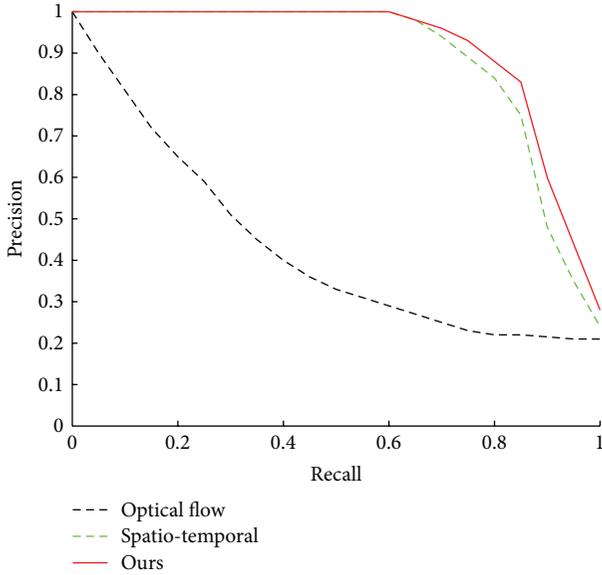


FIGURE 7: Performance curves on Boat-River dataset.

Actually, this alternative strategy is different from the one we proposed in the last subsection in the following perspectives. Firstly, the alternative one explicitly considers the predictability and the discriminability of the attributes and forces the model parameter to generate this kind of attributes, while the other one ignores this. Thus, intuitively, it can achieve better performance because its attributes and model are more effective. Secondly, the learning algorithm is more time consuming, especially in the first step (line 4), such that it may need offline learning while the other one just needs online learning because it is quite extremely efficient. Thirdly, the attribute learning is too precise, such that its generalization ability is weak. Therefore the background change may severely degrade the performance.

Consequently, the proposed alternative can be applied to the scenarios where the background change is slow such that the learned attributes and model can perform steadily in long time. Under such scenario, this alternative can achieve better performance. But we also need to point out that this alternative can only be applied to some specific scenarios such as indoor surveillance system where lighting conditions rarely change.

**3.5. Online Update.** Because the background appearance and motion are always changing in the surveillance video, such as lighting condition and weather, an anomaly detection system is expected to be self-adaptive to these changes, both in appearance and motion. Fortunately, the online update can be very efficient because of the ELM. Because the model parameters  $\mathbf{W}$  and  $\mathbf{B}$  are randomized, we actually do not need to change them. We just need to adjust  $\beta$ . In fact, the background change is usually slow during a short time (e.g., one second). So we just need to update  $\beta$  every second as follows.

In every second, we can randomly select some normal STVs (e.g., 10,000 to 20,000). We can obtain the low-level visual features of them and their relationship with each event. Here we need to adjust the relationship matrix  $\mathbf{E}$  first because it is the output of the ELM; thus all of its elements are not strictly 1 or  $-1$ . For each STV  $j$ , we can get the most related event as  $i' = \arg \max_i e_{ij}$ . Typically, we have  $e_{i'j} \approx 1$  but  $e_{i'j} \neq 1$ . So we can adjust  $e_{.j}$  as

$$e_{ij} = \begin{cases} 1, & \text{if } i = i' \\ -1, & \text{otherwise.} \end{cases} \quad (25)$$

Now we can get other training data  $\mathbf{H}$  and  $\mathbf{E}$  and we just need to retrain the ELM to update  $\beta$  given  $\mathbf{W}$  and  $\mathbf{B}$ . This procedure can be quite efficient because only simple linear operations are required; thus the update can be applied to online and real-time scenarios.

#### 4. Spatiotemporal Pyramid

Because of the presence of noise in surveillance video which is common in real-world applications, the judgement given by approach proposed above is sometimes wrong such that the detection performance is unsatisfactory for real-world application. Without postprocessing step, the detection may have low true positive rate but high false positive rate; that is, some anomalies are ignored while some normal STVs are misjudged as anomalous, which is unexpected for an effective anomaly detection approach where high true positive rate and low false positive rate are required. Fortunately, it is easy to observe that an anomalous event shows continuity in space and time; that is, it is always related to different parts in the camera and it may last for a period of time.



FIGURE 8: Experiments on Train dataset.

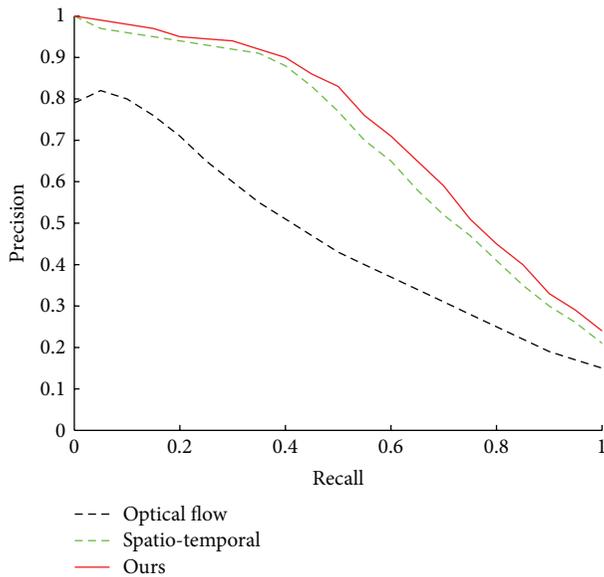


FIGURE 9: Performance curves on Train dataset.

Therefore, taking the spatial and temporal relationship of STVs into consideration can lead to more robust detection and significantly promote the detection performance. In this paper, we propose to use spatiotemporal pyramid (STP) to capture the relationship, as illustrated in Figure 1. We can use any levels based on the specific situation. In this paper we just use two-level STP, but we find that satisfactory result can be achieved under this setting.

Here we need to point out again that the HOG feature of upper level STV can be efficiently constructed from its lower-level STVs; thus the efficiency can be guaranteed. Actually, we can observe that the upper level STV can capture the spatial and temporal relationship of lower level STVs and the global information of an event. And because STVs in different levels of STP have different scales, our approach can perform multiscale detection based on STP. Given a STV in any scale (upper level or lower level), the anomaly judgement can be made by extreme leaning machine introduced above individually; that is, we have one ELM for each scale. Then we can combine the judgement for STVs in both upper and lower levels of STP and the judgement for neighbor STVs to give the finally refined judgement.

Before we give the judgement rules based on STP, there is an interesting and important phenomenon we need to mention, which is also essential for achieving satisfactory detection performance. The judgement on upper level STV tends to have high precision but low recall, implying that our approach can highly confidently claim that an upper level STV is anomalous, but some anomalous STVs may be missed. This is reasonable because the upper level STV can capture the global information of an event where the spatial and temporal continuity of an event can be adequately taken into consideration while some important local details will be ignored. On the other hand, the judgement on the lower level STV usually has low precision but high recall; that is, it tends to treat a STV as anomalous because it is too sensitive to local details and noise and ignores the spatial and temporal relationship between STVs, but it can capture more local information than upper level STV. Thus we propose the spatiotemporal pyramid to combine these two-level STVs to take advantage of both local details and global information simultaneously as follows.

On one hand, if it is judged to be abnormal for an upper STV, this judgement is highly confidential. If it is judged to be normal, it still has marked probabilistic to be anomalous. Thus the following results should be take into consideration too: (1) its six neighbors which consider the spatial and temporal continuity of events and (2) its lower STVs which capture the local detail information of events. To utilize all these pieces of information above, in this paper, an upper STV is finally judged to be anomalous if any of the following three criteria is satisfied: (1) it is judged to be anomalous, (2) at least three of its neighbors are anomalous, and (3) at least five of its lower level STVs are anomalous. Actually, the high-precision result for upper level STV leads to the first criterion. The second criterion is based on the spatial and temporal continuity of events. And the third criterion is based on a voting scheme because it is reasonable to assume that though one lower STV may be influenced by noise or local details, it is difficult for most STVs to generate wrong judgement.

On the other hand, lower-level STVs cannot capture the spatial and temporal continuity of events and they are significantly affected by noise in surveillance videos; consequently the judgement of a STV should be incorporated with its upper level STV and neighbors which consider the global information and the spatial temporal continuity of STVs. Therefore a lower-level STV is finally considered to be



FIGURE 10: Experiments on Ped1 dataset.

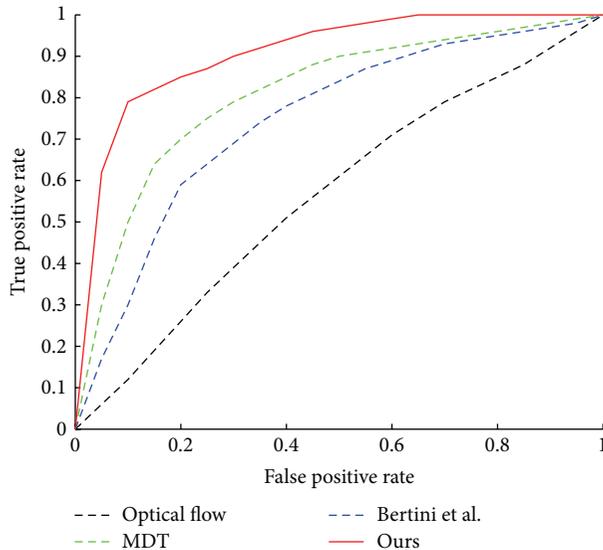


FIGURE 11: Performance curves on Ped1 dataset.

anomalous if it is judged to be anomalous and (1) two of more of its neighbors are anomalous or (2) its upper level STV is anomalous.

Based on the spatiotemporal pyramid and criteria above, we take into consideration the spatial and temporal continuity of events, the relationship between STVs in space and time, and the local details simultaneously which can significantly promote the performance. Furthermore, the spatiotemporal pyramid allows us to perform multiscale detection because STVs in different levels have different scales.

## 5. Experiment

To verify the effectiveness of the proposed approach, we test it in the following two publicly available datasets for anomaly detection: anomaly behavior detection dataset [53] (<http://www.cse.yorku.ca/vision/research/>) and UCSD pedestrian dataset [18] (<http://www.svcl.ucsd.edu/projects/anomaly>). The evaluation and comparison of different approaches are presented in two kinds of performance curves, that is, precision-recall, ROC curves, and equal error rate (EER) at both frame level and pixel level is also reported. As mentioned above, we use a two-level pyramid, and the size of lower level STV is  $10 \times 10 \times 10$ . To extract HOG features, we set  $n_\phi = 8$  and  $n_\theta = 16$ . We set the number of visual attributes,

that is, the number of hidden units in extreme learning machine, to 256, and the anomaly probabilistic  $p_a = 10^{-3}$ . In fact, our method does not require any training data because it is totally unsupervised. It just use the first one or two seconds of a test video to construct the initial normal events set. Furthermore, we set that the extreme learning machine is updated every one second. Furthermore, the following several state-of-the-art approaches for anomaly detection are compared to our approach: optical flow [15], Mahadevan et al. [18], sparse reconstruction (Cong et al.) [19], Zaharescu and Wildes [53], Reddy et al. [52], and Bertini et al. [20].

The first dataset is *Bellevue* dataset. It is a traffic scene where the lighting conditions change during the day gradually. Cars running from top to bottom or vice versa is normal event, while cars entering or exiting from the intersection from left or right and people in the lane are the anomalous events. The second is *Boat-River* dataset. The normal events are the waves in the river, while the anomalous event is defined as a boat that passing the scene. Actually, the boat is the newly observed object in the scene. The third is *Train* dataset. The normal events are people sitting on the train and some background change while anomalies are moving people. This dataset is quite challenging because the illumination varies drastically and the camera is not stable. The results on three datasets above, including the anomalous regions detected by our approach (highlighted in red) and the precision-recall curves of different approaches, are shown in Figures 4, 5, 6, 7, 8, and 9, respectively. We can observe that our approach is superior to state-of-the-art methods, for example, Zaharescu and Wildes. The superiority of our approach is based on the following three main reasons: (1) our approach address the semantic gap between low-level visual features and high-level events via visual attributes as intermediary such that more effective model can be constructed between them, (2) our approach can update model frequently (every one second) and quite efficiently such that it is very robust to drastic background change, (3) the spatial and temporal relationship between neighbor STVs is fully taken into consideration in our approach; therefore it is robust to local noise, and (4) our approach considers the spatial and temporal continuity of anomalous event; thus complicated events which may last several seconds or cover a large region can be effectively detected.

The UCSD dataset contains two subsets corresponding to two different scenes captured by fixed cameras overlooking the pedestrian walkways: Ped1 in which people with some distortion move towards and away from the camera and



FIGURE 12: Experiments on Ped2 dataset.

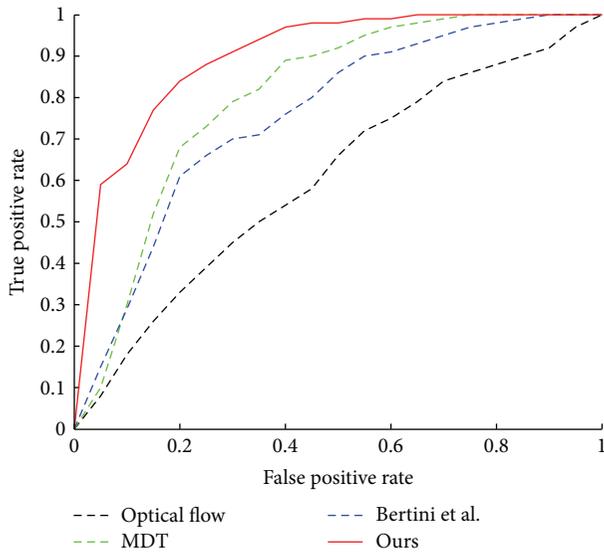


FIGURE 13: Performance curves on Ped2 dataset.

Ped2 which shows the pedestrian movement parallel to the camera. All videos are recorded with 10 frames per second. The resolution of Ped1 and Ped2 is  $238 \times 158$  and  $360 \times 240$ , respectively. Specifically, pedestrians walking in walkways are regarded as normal events, while nonpedestrian objects, for example, small carts, or nonwalking pedestrians, for example, cyclists and skaters, are treated as the anomalies. We test our approach on both subsets. We follow the evaluation adopted in [18, 20]. In the frame level, an anomalous frame is considered correctly detected if at least one pixel is detected as anomalous. In the pixel level, an anomalous frame is considered correctly detected only if at least 40% of the anomalous pixels are detected correctly. Because of the “lucky guess” which means the detected anomalies are different from the true anomalies in a frame, the frame level evaluation is sometimes not convincing enough because it does not take this phenomenon into consideration. Thus we adopt pixel level evaluation. We report the anomalous regions detected by our approach, the ROC curves, and the equal error rate (EER) which is the rate where the false positive rate is equal to 1 minus the true positive rate. The anomalous regions detected by our approach, the ROC curves, and the ERR of different approaches are shown in Figures 10, 11, 12, and 13 and Table 1, respectively. From the experiment results, we

TABLE 1: Comparison of the proposed approach and the state-of-the-art approaches for anomaly detection using Ped datasets. Approaches with \* can perform real-time detection.

|                       | Ped1        |             | Ped2        |             |
|-----------------------|-------------|-------------|-------------|-------------|
|                       | EER (frame) | EER (pixel) | EER (frame) | EER (pixel) |
| Optical flow* [15]    | 38%         | 76%         | 42%         | 80%         |
| Mahadevan et al. [18] | 25%         | 58%         | 25%         | 55%         |
| Cong et al. [19]      | 19%         | —           | 20%         | —           |
| Reddy et al.* [52]    | 22.5%       | 32%         | 21%         | 31%         |
| Bertini et al.* [20]  | 31%         | 70%         | 30%         | 68%         |
| Ours*                 | <b>17%</b>  | <b>28%</b>  | <b>16%</b>  | <b>26%</b>  |

can observe that our approach can significantly outperform the state-of-the-art approaches on both subsets for both frame level detection and pixel level detection, especially compared to real-time detection approaches, which validates the effectiveness of our approach for real-time anomaly detection in surveillance videos.

## 6. Conclusion

In this paper, a novel automatic anomaly detection approach is proposed. Densely sampled spatiotemporal video volumes represented by spatiotemporal local features are the fundamental of our approach. Normal event set is efficiently constructed from test data in an unsupervised way. We use visual attribute as intermediary to bridge the large semantic gap between low-level visual feature and high-level event. Extreme learning machine is utilized to model this three-level framework and it can be efficiently updated such that our approach is adaptive to background change. We propose to use spatiotemporal pyramid to capture the relationship between different STVs and the spatial and temporal continuity of anomalous events. Extensive experiments on several public datasets are conducted and the superior performance compared to several state-of-the-art approaches verifies the effectiveness of our approach.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## References

- [1] T. Troscianko, A. Holmes, J. Stillman, M. Mirmehdi, D. Wright, and A. Wilson, "What happens next? The predictability of natural behaviour viewed through CCTV cameras," *Perception*, vol. 33, no. 1, pp. 87–101, 2004.
- [2] H. Keval and M. A. Sasse, "'Not the usual suspects': a study of factors reducing the effectiveness of CCTV," *Security Journal*, vol. 23, no. 2, pp. 134–154, 2010.
- [3] N. Haering, P. L. Venetianer, and A. Lipton, "The evolution of video surveillance: an overview," *Machine Vision and Applications*, vol. 19, no. 5–6, pp. 279–290, 2008.
- [4] C. Brax, L. Niklasson, and M. Smedberg, "Finding behavioural anomalies in public areas using video surveillance data," in *Proceedings of the 11th International Conference on Information Fusion (FUSION '08)*, pp. 1–8, Cologne, Germany, July 2008.
- [5] I. Ivanov, F. Dufaux, T. M. Ha, and T. Ebrahimi, "Towards generic detection of unusual events in video surveillance," in *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS '09)*, pp. 61–66, Genoa, Italy, September 2009.
- [6] P. Antonakaki, D. Kosmopoulos, and S. J. Perantonis, "Detecting abnormal human behaviour using multiple cameras," *Signal Processing*, vol. 89, no. 9, pp. 1723–1738, 2009.
- [7] S. Calderara, C. Alaimo, A. Prati, and R. Cucchiara, "A real-time system for abnormal path detection," in *Proceedings of the IEEE 3rd International Conference on Imaging for Crime Detection and Prevention (ICDP '09)*, December 2009.
- [8] C. Liu, G. Wang, W. Ning, X. Lin, L. Li, and Z. Liu, "Anomaly detection in surveillance video using motion direction statistics," in *Proceedings of the 17th IEEE International Conference on Image Processing (ICIP '10)*, pp. 717–720, IEEE, Hong Kong, September 2010.
- [9] C. Piciarelli and G. L. Foresti, "Surveillance-oriented event detection in video streams," *IEEE Intelligent Systems*, vol. 26, no. 3, pp. 32–41, 2011.
- [10] C. C. Loy, T. Xiang, and S. Gong, "Detecting and discriminating behavioural anomalies," *Pattern Recognition*, vol. 44, no. 1, pp. 117–132, 2011.
- [11] R. R. Sillito and R. B. Fisher, "Semi-supervised learning for anomalous trajectory detection," in *Proceedings of the 19th British Machine Vision Conference (BMVC '08)*, September 2008.
- [12] C. Piciarelli and G. L. Foresti, "On-line trajectory clustering for anomalous events detection," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1835–1842, 2006.
- [13] O. Boiman and M. Irani, "Detecting irregularities in images and in video," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 17–31, 2007.
- [14] T. Xiang and S. Gong, "Video behavior profiling for anomaly detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 893–908, 2008.
- [15] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 555–560, 2008.
- [16] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 935–942, Miami, Fla, USA, June 2009.
- [17] F. Jiang, Y. Wu, and A. K. Katsaggelos, "A dynamic hierarchical clustering method for trajectory-based unusual video event detection," *IEEE Transactions on Image Processing*, vol. 18, no. 4, pp. 907–913, 2009.
- [18] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 1975–1981, San Francisco, Calif, USA, June 2010.
- [19] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pp. 3449–3456, June 2011.
- [20] M. Bertini, A. Del Bimbo, and L. Seidenari, "Multi-scale and real-time non-parametric approach for anomaly detection and localization," *Computer Vision and Image Understanding*, vol. 116, no. 3, pp. 320–329, 2012.
- [21] M. J. Roshtkhari and M. D. Levine, "Online dominant and anomalous behavior detection in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, 2012.
- [22] L. Kratz and K. Nishino, "Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR '09)*, pp. 1446–1453, Miami, Fla, USA, June 2009.
- [23] S. Khalid, "Activity classification and anomaly detection using  $m$ -medioids based modelling of motion patterns," *Pattern Recognition*, vol. 43, no. 10, pp. 3636–3647, 2010.
- [24] F. Jiang, J. Yuan, S. A. Tsafaris, and A. K. Katsaggelos, "Anomalous video event detection using spatiotemporal context," *Computer Vision and Image Understanding*, vol. 115, no. 3, pp. 323–333, 2011.
- [25] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [26] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR '09)*, pp. 1778–1785, Miami, Fla, USA, June 2009.
- [27] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 365–372, IEEE, Kyoto, Japan, October 2009.
- [28] M. Rastegari, A. Farhadi, and D. Forsyth, "Attribute discovery via predictable discriminative binary codes," in *Computer Vision—ECCV 2012: Proceedings of the 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Part VI*, vol. 7577 of *Lecture Notes in Computer Science*, pp. 876–889, Springer, Berlin, Germany, 2012.
- [29] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 2, pp. 2169–2178, June 2006.
- [30] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: unsupervised, multilevel, and long-term adaptive approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2287–2301, 2011.
- [31] K. Ouyirach, S. Gharti, and M. N. Dailey, "Incremental behavior modeling and suspicious activity detection," *Pattern Recognition*, vol. 46, no. 3, pp. 671–680, 2013.

- [32] Y. Benezeth, P.-M. Jodoin, and V. Saligrama, "Abnormality detection using low-level co-occurring events," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 423–431, 2011.
- [33] T. Hospedales, S. Gong, and T. Xiang, "Video behaviour mining using a dynamic topic model," *International Journal of Computer Vision*, vol. 98, no. 3, pp. 303–323, 2012.
- [34] Y. Benezeth, P.-M. Jodoin, V. Saligrama, and C. Rosenberger, "Abnormal events detection based on spatio-temporal co-occurrences," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops '09)*, pp. 2458–2465, Miami, Fla, USA, June 2009.
- [35] E. B. Ermiş, V. Saligrama, P.-M. Jodoin, and J. Konrad, "Motion segmentation and abnormal behavior detection via behavior clustering," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '08)*, pp. 769–772, IEEE, San Diego, Calif, USA, October 2008.
- [36] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.
- [37] A. Mittal, A. Monnet, and N. Paragios, "Scene modeling and change detection in dynamic scenes: a subspace approach," *Computer Vision and Image Understanding*, vol. 113, no. 1, pp. 63–79, 2009.
- [38] X. Zhu and Z. Liu, "Human behavior clustering for anomaly detection," *Frontiers of Computer Science in China*, vol. 5, no. 3, pp. 279–289, 2011.
- [39] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.
- [40] T. M. Hospedales, J. Li, S. Gong, and T. Xiang, "Identifying rare and subtle behaviors: a weakly supervised joint topic model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 12, pp. 2451–2464, 2011.
- [41] J. Li, S. Gong, and T. Xiang, "Learning behavioural context," *International Journal of Computer Vision*, vol. 97, no. 3, pp. 276–304, 2012.
- [42] E. Ricci, G. Zen, N. Sebe, and S. Messelodi, "A prototype learning framework using EMD: application to complex scenes analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 513–526, 2012.
- [43] P.-M. Jodoin, J. Konrad, and V. Saligrama, "Modeling background activity for behavior subtraction," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '08)*, pp. 1–10, Stanford, Calif, USA, September 2008.
- [44] P. Jodoin, V. Saligrama, and J. Konrad, "Behavior subtraction," *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 4244–4255, 2012.
- [45] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM International Conference on Multimedia (MULTIMEDIA '07)*, pp. 357–360, September 2007.
- [46] Y. Nesterov, "Gradient methods for minimizing composite objective function," CORE Discussion Papers no. 2007-76, CORE, 2007.
- [47] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on bow framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, June 2014.
- [48] J. W. Cao and Z. Lin, "Bayesian signal detection with compressed measurements," *Information Sciences*, vol. 289, pp. 241–253, 2014.
- [49] Y. Jin, J. Cao, Q. Ruan, and X. Wang, "Cross-modality 2D-3D face recognition via multiview smooth discriminant analysis based on ELM," *Journal of Electrical and Computer Engineering*, vol. 2014, Article ID 584241, 9 pages, 2014.
- [50] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*, Wiley-Interscience, 1971.
- [51] D. Serre, *Matrices: Theory and applications*, Springer, 2002.
- [52] V. Reddy, C. Sanderson, and B. C. Lovell, "Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '11)*, pp. 55–61, Colorado Springs, Colo, USA, June 2011.
- [53] A. Zaharescu and R. Wildes, "Anomalous behaviour detection using spatiotemporal oriented energies, subset inclusion histogram comparison and event-driven processing," in *Computer Vision—ECCV 2010*, pp. 563–576, 2010.

## Research Article

# Improving ELM-Based Service Quality Prediction by Concise Feature Extraction

**Yuhai Zhao, Ying Yin, Gang Sheng, Bin Zhang, and Guoren Wang**

*College of Information Science and Engineer, Northeastern University, Shenyang 110819, China*

Correspondence should be addressed to Yuhai Zhao; zhaoyuhai@ise.neu.edu.cn

Received 20 August 2014; Revised 10 November 2014; Accepted 12 November 2014

Academic Editor: Jiuwen Cao

Copyright © 2015 Yuhai Zhao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Web services often run on highly dynamic and changing environments, which generate huge volumes of data. Thus, it is impractical to monitor the change of every QoS parameter for the timely trigger precaution due to high computational costs associated with the process. To address the problem, this paper proposes an active service quality prediction method based on extreme learning machine. First, we extract web service trace logs and QoS information from the service log and convert them into feature vectors. Second, by the proposed EC rules, we are enabled to trigger the precaution of QoS as soon as possible with high confidence. An efficient prefix tree based mining algorithm together with some effective pruning rules is developed to mine such rules. Finally, we study how to extract a set of diversified features as the representative of all mined results. The problem is proved to be NP-hard. A greedy algorithm is presented to approximate the optimal solution. Experimental results show that ELM trained by the selected feature subsets can efficiently improve the reliability and the earliness of service quality prediction.

## 1. Introduction

The advantage of composite Web services is that it realizes a complex application by connecting multiple component services seamlessly. However, in real applications, Web service lives in a highly dynamic environment, and both the network condition and the operational status of each of the component Web services (WSs) may change during the lifetime of a business process itself. The instability brought by various uncertain factors often makes the composite services failed or interrupted temporally. Therefore, it is very important to ensure the normal execution of the composite service applications and provide a reliable software system [1].

As one of the promising technologies to address the above issue, Web service quality prediction has become an important research problem and has attracted a lot of attention in recent years. The goal is to perceive in advance whether the invoked services will fail or be interrupted by monitoring and evaluating service quality fluctuation. In SOA infrastructure, Web service prediction aims to optimally select the high quality service in advance to ensure the reliable execution of system. A number of Web service prediction

models have been proposed, such as ML-based methods [2–4], QoS-aware based methods [5], and collaborative filtering-based methods [6, 7]. These models are often implemented by monitoring and evaluating the quality of composite services. In spite of improving the quality of composite services to some extent, these methods still have three major drawbacks. First, most of the traditional ML-based prediction models [3], such as support vector machines (SVM) and artificial neural networks (ANN), are more sensitive to the user-specified parameters. Second, the prediction models based on QoS monitoring, such as Naive Bayes and Markov model [8, 9], often assume that sequences in a class are generated by an underlying model  $M$  and the probability distributions are described by a set of parameters. However, these parameters are obtained by predicting QoS during the whole lifecycle of the services and will therefore lead to high overhead costs. In another sequence distance based prediction method, such as collaborative filtering [6, 7], a function measuring the similarity between a pair of sequences is necessary. However, how to select an optimal similarity function is far from trivial, as it will introduce numerous parameters and measures for distances which may be rather subjective.

As a powerful prediction model, extreme learning machine (ELM for short) was originally developed based on single-hidden layer feedforward neural networks (SLFNs) in [10]. Compared with the conventional learning machines, it is of extremely fast learning capacity and good generalization capability. Thus, ELM, with its variants, has been widely applied in many fields. For example, in [11], ELM was applied for plain text classification by using the one-against-one (OAO) and one-against-all (OAA) decomposition scheme. In [12], an ELM-based XML document classification framework was proposed to improve classification accuracy by exploiting two different voting strategies. A protein secondary structure prediction framework based on ELM was proposed in [13, 14] to provide good performance at extremely high speed. References [15, 16] evaluated the multicategory classification performance of ELM on three microarray datasets. The results indicate that ELM produces comparable or better classification accuracies with reduced training time and implementation complexity compared to artificial neural networks methods and support vector machine methods.

In this paper, we introduce ELM into Web service QoS prediction. To our best knowledge, it has never been addressed by any previous work. However, it is not trivial to integrate ELM into Web services quality prediction. Some issues need further consideration, for example, how to model the execution information of Web services to facilitate the usage of ELM on the data and how to train ELM in as short time as possible to get a model of high prediction accuracy so that we can conduct an on-line Web service QoS prediction.

Our contributions include that (1) we devise a method to extract web service trace logs and QoS information from the service log and convert them into feature vectors; (2) we propose a concept, namely, EC rule, based on which we are enabled to trigger precaution as soon as possible with high confidence; (3) we develop an efficient prefix tree based mining algorithm together with some effective pruning rules to mine such rules; (4) we further study how to extract a set of diversified features as the representative of all mined results based on ELM.

The rest of this paper is organized as follows. Section 2 gives a brief overview of ELM. Section 3 presents ELM-based QoS prediction framework. Section 4 studies the feature vectors representation of Web services. Section 5 defines the EC rules and proposes the mining algorithm. In Section 6, we study the problem of diversified feature selection and present the greedy solution. In Section 7, the experimental evaluation results are reported. Finally, Section 8 concludes this paper.

## 2. A Brief Introduction to ELM

ELM (extreme learning machine) is a generalized single hidden-layer feedforward network. In ELM, the hidden-layer node parameter is mathematically calculated instead of being iteratively tuned; thus, it provides good generalization performance at thousands of times faster speed than traditional popular learning algorithms for feedforward neural networks [12].

Given  $N$  arbitrary distinct samples  $(x_i, t_i)$ , where  $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T \in \mathbf{R}^n$  and  $t_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbf{R}^m$ , standard SLFNs with  $L$  hidden nodes and activation function  $g(x)$  are mathematically modeled as

$$f(x) = \sum_{i=1}^L \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}), \quad (1)$$

where  $\mathbf{a}_i$  and  $b_i$  are the learning parameters of hidden nodes and  $\beta_i$  is the weight connecting the  $i$ th hidden node to the output node.  $g(\mathbf{a}_i, b_i, \mathbf{x})$  is the output of the  $i$ th hidden node with respect to the input  $x$ . In our case, sigmoid type of additive hidden nodes is used. Thus, (1) is given by

$$f(x) = \sum_{i=1}^L \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}) = \sum_{i=1}^L \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad (2)$$

$$(j = 1, \dots, N),$$

where  $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{im}]^T$  is the weight vector connecting the  $i$ th hidden node and the input nodes,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the weight vector connecting the  $i$ th hidden node and the output nodes,  $b_i$  is the bias of the  $i$ th hidden node, and  $\mathbf{o}_j$  is the output of the  $j$ th node [10].

If an SLFN with activation function  $g(x)$  can approximate the  $N$  given samples with zero errors that  $\sum_{j=1}^L \|\mathbf{o}_j - \mathbf{t}_j\| = 0$ , there exist  $\beta_i$ ,  $\mathbf{a}_i$ , and  $b_i$  such that

$$\sum_{i=1}^L \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \dots, N. \quad (3)$$

Equation (3) can be expressed compactly as follows:

$$\mathbf{H}\beta = \mathbf{T}, \quad (4)$$

where

$$\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_L, b_1, \dots, b_L, \mathbf{x}_1, \dots, \mathbf{x}_N)$$

$$= \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L}, \quad (5)$$

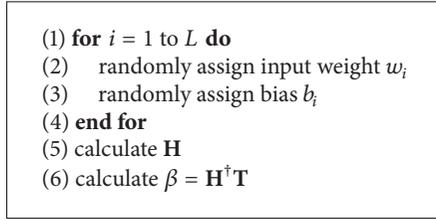
$$\beta = [\beta_1^T, \dots, \beta_L^T]_{m \times L}^T, \quad \mathbf{T} = [\mathbf{t}_1^T, \dots, \mathbf{t}_L^T]_{m \times N}^T.$$

$\mathbf{H}$  is called the hidden layer output matrix of the network. The  $i$ th column of  $\mathbf{H}$  is the  $i$ th hidden nodes output vector with respect to inputs  $x_1, x_2, \dots, x_N$  and the  $j$ th row of  $\mathbf{H}$  is the output vector of the hidden layer with respect to input  $x_j$ .

For the binary classification applications, the decision function of ELM [17] is

$$f(x) = \text{sign} \left( \sum_{i=1}^L \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}) \right) = \text{sign}(\beta \cdot h(x)). \quad (6)$$

$h(\mathbf{x}) = [g(\mathbf{a}_1, b_1, \mathbf{x}), \dots, g(\mathbf{a}_L, b_L, \mathbf{x})]^T$  is the output vector of the hidden layer with respect to the input  $\mathbf{x}$ .  $h(\mathbf{x})$  actually



ALGORITHM 1: ELM.

maps the data from the  $d$ -dimensional input space to the  $L$ -dimensional hidden layer feature space  $\mathbf{H}$ .

In ELM, the parameters of hidden layer nodes, that is,  $w_i$  and  $b_i$ , can be chosen randomly without knowing the training datasets. The output weight  $\mathbf{L}$  is then calculated with matrix computation formula  $\mathbf{L} = \mathbf{H}^\dagger \mathbf{T}$ , where  $\mathbf{H}^\dagger$  is the Moore-Penrose inverse of  $\mathbf{H}$ .

ELM tends to reach not only the smallest training error but also the smallest norm of weights [18]. Given a training set  $\mathcal{N} = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \dots, N\}$ , activation function  $g(x)$ , and hidden node number  $L$ , the pseudocode of ELM [10] is given in Algorithm 1.

### 3. The ELM-Based QoS Prediction Framework

In order to immediately comprehend our idea, we illustrate the whole process of ELM-based Web service QoS prediction shown in Figure 1. As shown, the process consists of four major phases: (1) preprocess, which records the composite service execution log information, extracts multidimensional QoS attributes, and converts them into service feature vectors; (2) the EC rules mining, where a prefix tree based algorithm is proposed to mine the candidate feature sets, namely, the EC rules; (3) diversified feature selection, where a small subset of diversified features are extracted from all the rules to construct a classifier of high prediction accuracy, that is, F-ELM; (4) feature updating, where the process periodically updates the prefix tree with the QoS values changing.

(1) *Preprocess*. At first, the system needs to collect large amounts of composite service execution information, aiming to mine useful knowledge for prediction. The original service log includes a variety of structural and unstructural data information, such as service trace logs, quality of service (QoS) information, service invocation relationships, and Web service description language (WSDL). These sets of information are typically heterogeneous, of multiple data types, and high dynamic. Thus, in order to extract the useful feature vectors, a preprocess step is necessary. This part will be discussed in Section 4.

(2) *The EC Rules Mining*. Since the goal is to conduct an online Web service QoS prediction, the rules should be concise so as to response the predictor as early as possible. By the proposed EC rules, we are enabled to trigger the prediction as soon as possible with high confidence. An efficient prefix tree based mining algorithm together with some effective

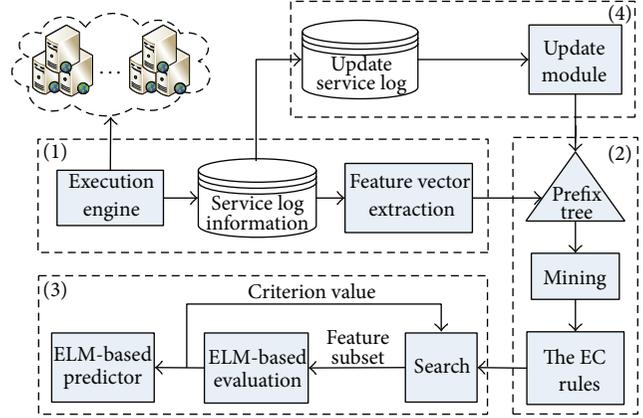


FIGURE 1: ELM-based QoS prediction process.

pruning rules is developed to mine such rule. This part will be described in Section 5.

(3) *Diversified Feature Selection*. Too many rules increase the chance for model overfitting and decrease the generalization performance of a model. Thus, in this step, we study how to extract a small subset of diversified features as the representative of all mined results. By an ELM-based evaluation, the feature subset of the highest score is utilized to construct the predictor, that is, F-ELM. This part will be described in Section 6.

(4) *Features Updating*. Further, when a new service sequence is input, the update module judges QoS status of the service sequences. If the status of a service attribute changes greatly, the update module sends the updating request to the prefix tree according to a certain strategy. Besides judging the status, the feature values of each node in the prefix tree are recalculated periodically. In this paper, we exploit the strategy mentioned in [19] to address the issue.

In what follows, we mainly focus on steps (1)~(3) one by one.

### 4. Preprocessing

Once a service-oriented application or a composite service is deployed in a running environment, the application can be executed in many execution instances. Each execution instance is uniquely identified with an identifier (i.e., id). In each execution instance, a set of service components can be triggered. Due to various internet uncertain factors, there possibly exist a large number of sets of potential exception status information. We record the triggered events of the Web service failure information in a log. It is helpful for service quality management by extracting execution status information from the execution log to predict the service reliability.

Web service QoS information often includes many attributes. For example, the literature [20] lists twelve attributes to depict service QoS, for example, response time,

TABLE 1: Composite QoS status information.

| A  | Composite QoS                     | Description                                       |
|----|-----------------------------------|---------------------------------------------------|
| 0  | $\langle av^0, exe^0 \rangle$     | Server unavailable, runtime delay                 |
| 1  | $\langle av^{0.5}, exe^0 \rangle$ | Server available intermittently, runtime delay    |
| 2  | $\langle av^1, exe^0 \rangle$     | Server available, runtime delay                   |
| 3  | $\langle av^{0.5}, exe^1 \rangle$ | Server available intermittently, normal execution |
| 4  | $\langle av^1, exe^1 \rangle$     | Server available, normal execution                |
| 5× | $\langle av^0, exe^1 \rangle$     | Server unavailable, normal execution              |

availability, throughput, successability, reliability, compliance, latency, service name, WSDL address, documentation, and service classification. To simplify the explanation, we assume that there are just two QoS attributes for each component service in this paper, that is, the availability attribute (av) and the execution time attribute (exe). We further suppose that there are three possible states for av, that is, inaccessible, intermittently accessible, and accessible, denoted by  $av^0$ ,  $av^{0.5}$ , and  $av^1$ , respectively, and two states for exe, that is, delayed execution and normal execution, denoted by  $exe^0$  and  $exe^1$ , respectively. As such, we obtain five possible groups of service execution statuses as shown in Table 1:  $\langle av^0, exe^0 \rangle$ , denoted by  $S^0$ , corresponding to the status of server unavailable and runtime delay;  $\langle av^{0.5}, exe^0 \rangle$ , denoted by  $S^1$ , the status of server available intermittently and runtime delay;  $\langle av^1, exe^0 \rangle$ , denoted by  $S^2$ , the status of server available and runtime delay;  $\langle av^{0.5}, exe^1 \rangle$ , denoted by  $S^3$ , the status of server available intermittently but normal execution; and  $\langle av^1, exe^1 \rangle$ , denoted by  $S^4$ , the status of server available but normal execution. Note that the status  $\langle av^0, exe^1 \rangle$ , denoted by  $S^5$ , does not exist in practice. This is because an unavailable service is not executed.

Given the QoS status representation in Table 1, we extract Web service trace logs and QoS information from the service log and convert them into feature vectors by the following way. Let  $S = \{S_1, S_2, \dots, S_n\}$  be the candidate service component set and  $S_i^j$  the status  $j$  of service  $S_i$ . For every record in the web service log, we replace each individual component service by the corresponding status such that every record could be converted into a sequence of feature vectors. Table 2 exemplifies a service execution dataset of 15 failed executions, 5 successful executions, and 3 failure types. For example, column 2 in row 5 denotes an execution sequence “ $S_1^4 S_2^2 S_3^2 S_4^2$ ” which first invokes service  $S_1$  of the status  $s_4$  and then service  $S_2$  of the status  $s_2$ , service  $S_3$  of the status  $s_2$ , and service  $S_4$  of the status  $s_2$ . Column 3 indicates that the sequence was executed twice, and column 4 shows that this execution failed with error type C.

## 5. The EC Rules Mining

In the last step, we have modeled the data as a sequence dataset. Next, we detail how to mine the candidate features for on-line Web service QoS prediction from the sequence

TABLE 2: An example of service execution instances.

| ID | Execute log entry         | Count | Class              |
|----|---------------------------|-------|--------------------|
| 1  | $S_1^0$                   | 3     | Failed with type A |
| 2  | $S_1^1 S_3^1$             | 3     | Failed with type B |
| 3  | $S_1^2 S_2^2 S_3^2$       | 2     | Failed with type C |
| 4  | $S_1^1 S_2^1 S_3^1$       | 3     | Failed with type B |
| 5  | $S_1^4 S_2^2 S_3^2 S_4^2$ | 2     | Failed with type C |
| 6  | $S_1^4 S_2^2 S_3^2$       | 2     | Failed with type C |
| 7  | $S_1^4 S_2^3 S_3^4 S_4^2$ | 2     | Successful         |
| 8  | $S_1^4 S_2^3 S_3^4$       | 3     | Successful         |

dataset. Different from the sequence feature used in other domains, we require that the features in the context of on-line Web service QoS prediction should be of two important properties: (1) sequential character and (2) conciseness. This is because on-line Web service QoS prediction is a temporal process, where the prediction should be triggered as soon as possible.

*5.1. Basic Definition.* In this section, we first give some basic concepts and the problem statement.

*Definition 1 (feature).* Let  $\mathcal{D}$  be service execution log with service set  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ . Let  $\mathcal{F} = \{S_1^i S_2^j \dots S_l^k\} \subseteq \mathcal{S}$  ( $l = 1, 2, \dots, n$ ) be a component service or a subset of execution sequences containing status information. We call a set of component services with status information a Feature. Note that  $S_i^k$  is the status  $k$  of service  $S_i$ .

Given a feature  $\mathcal{F} = s_1^i s_2^j \dots s_m^k$ , we say feature  $\mathcal{F}$  appears in a sequence  $\mathcal{S}$  if there exists  $1 \leq i_0 \leq n - m + 1$  such that  $\mathbf{a}_{i_0} = s_1^i$ ,  $\mathbf{a}_{i_0+1} = s_2^j$ ,  $\dots$ ,  $\mathbf{a}_{i_0+m-1} = s_m^k$ . However, a feature  $\mathcal{F}$  may appear several times in a sequence. For example,  $\mathcal{F} = \{acd\}$  appears twice in sequence  $\mathcal{S} = \{aacdf gacde\}$ . Below, we give the minimum prefix length definition.

*Definition 2 (minimum prefix length).* Given feature  $\mathcal{F}$  and a sequence  $\mathcal{S}$ , where  $\mathcal{F} \subseteq \mathcal{S}$ , the minimum prefix length is the length from initial position of  $\mathcal{S}$  to the first matched position of  $\mathcal{F}$  (MPL( $\mathcal{F}, \mathcal{S}$ ) for short).

*Definition 3 (weight Intra\_Support with early factor).* Let  $\mathcal{F}$  be a feature and let  $C_k$  be a class. The weight intraclass support with early factor of feature  $\mathcal{F}$  in class  $C_k$  is the ratio of the sum of the reciprocals of the minimum prefix lengths containing  $\mathcal{F}$  in  $C_k$  to the number of data in class  $C_k$ .  $wis^e$  is an abbreviation of weight *Intra\_Support* with early factor. Consider

$$\begin{aligned}
 & wis^e(\mathcal{F} \rightarrow C_k) \\
 &= \frac{\sum_{\mathcal{S} \in \mathcal{S}(\mathcal{F} \rightarrow C_k)} (1/\text{MPL}(\mathcal{F}, \mathcal{S}(\mathcal{F} \rightarrow C_k)))}{\|C_k\|}. \quad (7)
 \end{aligned}$$

**Input:** data set  $D$ ,  $\alpha$  (minimum  $\text{wis}^e$ ),  $\beta$  and  $\alpha$   
**Output:** Feature Sets (FS)

- (1) Set FS =  $\phi$
- (2) Count support of 1-features in every class
- (3) Generate 1-feature set( $F_1$ )
- (4) Count support of 1-features in different class
- (5) Select 1-features respectively and add them to FS
- (6) new feature set  $\leftarrow$  Generate(2-feature set( $F_2$ ))
- (7) **while** new feature set is not empty **do**
- (8) Count  $\text{wis}^e(\mathcal{F}, C_k)$  of candidates in new feature set
- (9) For each feature  $\mathcal{F}$  in  $(l + 1)$ -feature set
- (10) **Applying pruning 1:** IF  $\text{wis}^e(\mathcal{F} \rightarrow C_k) < \alpha$
- (11) remove feature  $\mathcal{F}$ ;
- (12) Else if there is a superset  $\mathcal{F}a$  of feature  $\mathcal{F}$  in  $l$ -feature set
- (13) **Applying pruning 2:** that  $\text{wis}^e(\mathcal{F}a) = \text{wis}^e(\mathcal{F})$  or
- (14) **Applying pruning 3:**  $\log((\sigma + \text{wis}^e(\mathcal{F} \rightarrow C_k))/(\sigma + \text{wis}^e(\mathcal{F} \rightarrow \neg C_k))) \geq \beta$
- (15) Then remove feature  $\mathcal{F}$ ;
- (16) Select optimal features to FS;
- (17) **ENDIF**
- (18) **end while**
- (19) new feature set  $\leftarrow$  Generate(next level features sets)
- (20) Return FS;

**Function 1 Generate( $l + 1$ )-feature Set**

- (21) Let  $(l + 1)$ -feature set be empty set
- (22) (Note: Obey by the  $\text{CI}_{k-1} * \text{CI}_{k-1}$  Method to Merge)
- (23) **for** each pair of features  $pP_{l-1}$  and  $P_{l-1}q$  in  $l$ -feature set **do**
- (24) Insert candidate  $P_{l-1} \cdot pq$  in  $(l + 1)$ -feature set;
- (25) **for** all  $P_l \subset P_{l-1}pq$  **do**
- (26) **if**  $P_l$  does not exist in  $l$ -feature set **then**
- (27) Then remove candidate  $P_{l-1}pq$
- (28) **end if**
- (29) Return  $(l + 1)$ -feature set
- (30) **end for**
- (31) **end for**

ALGORITHM 2: The EC-Miner algorithm.

$\text{wis}^e(\mathcal{F} \cup C_k)$  denotes the support of features  $\mathcal{F}$  and  $C_k$  emerging simultaneously,  $\text{wis}^e(\mathcal{F} \cup \neg C_k)$  denotes the support of features  $\mathcal{F}$  and  $\neg C_k$  emerging simultaneously; that is,  $\text{wis}^e(\mathcal{F} \cup \neg C_k) = \text{wis}^e(\mathcal{F}) - \text{wis}^e(\mathcal{F} \cup C_k)$ . We say that  $\text{wis}^e(\mathcal{F})$  is frequent if  $\text{wis}^e(\mathcal{F}) \geq \alpha$ , where  $\alpha$  is user-specific minimum frequent threshold.

**Definition 4 (discriminative feature).** Let  $\mathcal{F}$  be a feature and let  $C_k$  be a class. The discriminative power  $\mathcal{F}$ , denoted by DF, is calculated as follows:

$$\text{DF}(\mathcal{F}) = \log \left( \frac{\sigma + \text{Supp}(\mathcal{F} \rightarrow C_k)}{\sigma + \text{Supp}(\mathcal{F} \rightarrow \neg C_k)} \right), \quad (8)$$

where  $\sigma$  is a regulation factor. Specially, we say that  $\mathcal{F}$  is discriminate feature if  $\text{DF}(\mathcal{F}) \geq \beta$ , where  $\beta$  is a user-specific minimum discriminative threshold.

The rationale behind Definition 4 is intuitive. If a feature  $\mathcal{F}$  often occurs in class  $C_k$  but rarely in other classes (i.e.,  $\neg C_k$ ), we consider it a feature well discriminating  $C_k$  from the other classes. Moreover, since  $\text{Supp}(\mathcal{F} \rightarrow \neg C_k)$  may be zero, we add a regulation factor  $\sigma$  to avoid this case.

**Definition 5 (concise feature).** Given a specific class label  $C_k$  and a discriminate feature  $\mathcal{F}$ , the discriminative power of  $\mathcal{F}$  is no less than that of a longer feature  $\mathcal{F}'$  and  $\text{conf}(\mathcal{F}' \rightarrow C_k) \geq \text{conf}(\mathcal{F} \rightarrow C_k)$ , we say that  $\mathcal{F}$  is concise with respect to  $C_k$ , where  $\text{conf}(\mathcal{F} \rightarrow C_k) = \text{wis}^e(\mathcal{F}C_k)/\text{wis}^e(\mathcal{F})$ .

Definition 5 is also understandable. This is because if we have a shorter feature  $\mathcal{F}'$ , the discriminative power of which is no less than that of a longer feature  $\mathcal{F}$  such that  $\mathcal{F}' \sqsubseteq \mathcal{F}$ , there is no need to use  $\mathcal{F}$  instead of  $\mathcal{F}'$  for classification. That is, we prefer a feature of shorter size but stronger discriminative power. In this sense, we refer to such a feature as a concise feature.

**Problem Statement.** Given a Web service execution log  $D$ , a minimum frequent threshold  $\alpha$ , a regulation factor  $\sigma$ , and a minimum discriminative threshold  $\beta$ , our goal is to find all sequence rules satisfying both Definitions 3 and 5, that is, the EC rules.

**5.2. The EC-Miner Algorithm.** In this section, we detail the proposed EC rule mining algorithm, namely, EC-Miner. The main idea is formalized in Algorithm 2.

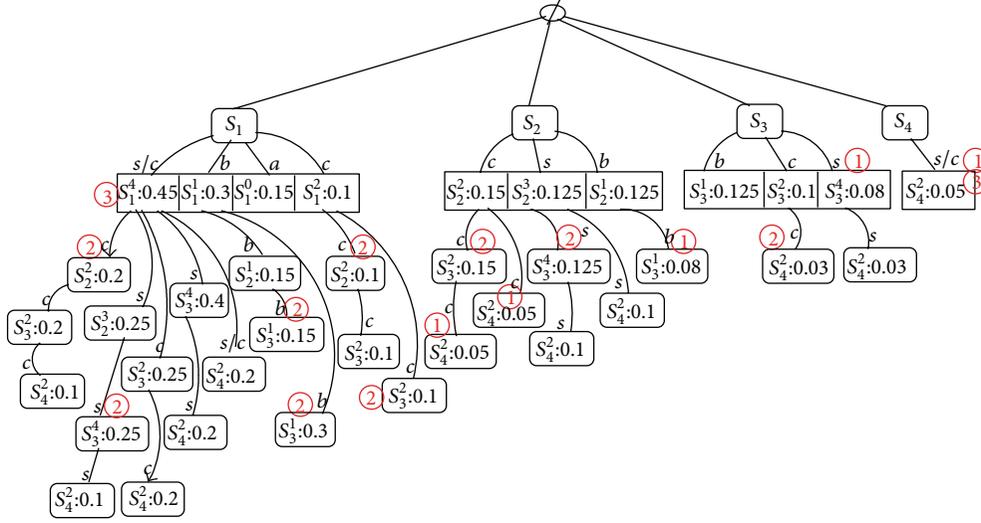


FIGURE 2: The EC rule mining.

The mining process is exemplified by a prefix tree as shown in Figure 2, which is built on Table 2 with  $wis^e = 0.1$ . As seen from Figure 2, there are four services  $s_1$ ,  $s_2$ ,  $s_3$ , and  $s_4$  at level 1. We obtain different status information for each service in descending order at level 2. Different from the traditional support computing method, we consider both concise and early characteristics. Therefore, the obtained order using  $wis^e$  for each item is also different from traditional approaches. For example, at first, the algorithm scans Table 2 once and computes the  $wis^e$  of each item. After computing, we generate the candidate early feature for the second level. We can see 11 candidate 1-features at level 2. The  $wis^e$  of each one item is  $s_1^4 : 0.45$ ,  $s_1^1 : 0.3$ ,  $s_1^0 : 0.15$ ,  $s_2^2 : 0.15$ ,  $s_3^3 : 0.125$ ,  $s_3^1 : 0.125$ ,  $s_2^2 : 0.1$ ,  $s_3^3 : 0.1$ , and so on, where the number after colon denotes weight of *Intra\_Support* with early factor ( $wis^e$ ). Next, we generate the candidate early 2-features for the third level. For example,  $s_1^4 s_3^3 : 0.25$  denotes the  $wis^e$  of  $s_1^4 s_3^3$  which is 0.25. The feature  $\mathcal{F}$  with solid box represents the corresponding rule. For example, class  $s$  with solid box under  $s_1^4 s_3^3$  means the rule  $s_1^4 s_3^3 \rightarrow \text{successful}$  can be deduced.

We can use the  $wis^e$ -based pruning 1 (see the details in Section 5.3) to prune some rules (lines 2–5). We can prune some redundancy rules by applying pruning rule 1 (lines 8–10); for example, candidates  $s_2^1$ ,  $s_4^4$ ,  $s_3^2 s_4^2$ ,  $s_3^4 s_4^2$  are removed since  $wis^e(s_3^1) = 0.08 < \alpha = 0.1$ ,  $wis^e(s_4^1) = 0.05 < \alpha = 0.1$ ,  $wis^e(s_3^2 s_4^2) = 0.03 < \alpha = 0.1$ ,  $wis^e(s_3^4 s_4^2) = 0.03 < \alpha = 0.1$ . In Figure 2, the pruning rule 1 is applied which is marked by ①. Further, if all features under threshold are pruned, the rules containing these features will be pruned.

Then, we perform the concise-based pruning rule 2 (lines 2–5, Algorithm 2), which will also be explained in Section 5.3. For instance,  $(s_1^2 s_2^2)$  in candidate  $(s_2^2 s_2^2; c)$  is terminated because  $wis^e(s_1^2) = wis^e(s_1^2 s_2^2) = 0.1$ . In Figure 2, the pruning rule 2 is applied which is marked by ②.

At last but not the least, discriminative-based pruning rule 3 is very important but not difficult to be understood (see Section 5.3). For example, candidate feature  $s_4^1$  is removed by line 14 because  $wis^e(s_4^1 \rightarrow c) = wis^e(s_4^1 \rightarrow \text{successful})$ . In Figure 2, the pruning rule 3 is applied which is marked by ③. A complete pseudocode for mining optimal EF sets is presented in Algorithm 2.

Algorithm 2 discusses the  $wis^e$ -based pruning, concise-based pruning, and discriminative-based pruning. Most of the existing algorithms find an interesting rule set by post-pruning. However, this may be very inefficient especially when the minimum support is low since it will generate an amount of redundancy rules. Our EC-Miner algorithm makes use of the interestingness measure property to efficiently prune uninteresting rules and saves only the maximal interesting rules instead of all ones. This distinguishes it from other association rule mining algorithms.

Function 1 is a function to generate candidate item sets. All generated candidates are built on the prefix tree structure. We adopt the  $CI_{k-1} * CI_{k-1}$  merge strategy [21] to obtain the candidate item sets. After rules have been formed, we can prune many redundancy rules.

**5.3. The Pruning Strategies.** To improve the efficiency of EC-Miner, we devise a series of pruning rules.

**Pruning Rule 1.** In pruning by  $wis^e$ : given  $wis^e$ , a feature  $\mathcal{F}$  and all its possible proper supersets  $\mathcal{F}a$ , and class  $C_k$ , if  $0 \leq wis^e(\mathcal{F} \rightarrow C_k) \leq \alpha$ , then  $\mathcal{F} \rightarrow C_k$  and  $\mathcal{F}a \rightarrow C_k$  are all not the EC rules.

*Proof.* Once  $0 \leq wis^e(\mathcal{F} \rightarrow C_k) \leq \alpha$  is observed, it is not necessary to search for more specific rules  $\mathcal{F}a \rightarrow C_k$ . Because  $wis^e(\mathcal{F}a \rightarrow C_k) \leq wis^e(\mathcal{F} \rightarrow C_k) \leq \alpha$ . So, target  $C_k \in C$  will be terminated in candidate rule  $\mathcal{F} \rightarrow C_k$ .  $\square$

Instead of the global support, pruning 1 describes the intraclass support of a feature with respect to a specific class. This is because a feature  $\mathcal{F}$  in  $C_k$  is hardly frequent if  $C_k$  is rare in service execution log. Thus, pruning rule 1 can reduce the redundancy rules greatly. This is different from association rules.

*Pruning Rule 2.* In pruning by conciseness, given  $\text{wis}^e$ , a feature  $\mathcal{F}$  and all its possible proper supersets  $\mathcal{F}a$ , and class  $C_k$ , if  $\text{wis}^e(\mathcal{F}) = \text{wis}^e(\mathcal{F}a)$ , then feature  $\mathcal{F}a$  and all its proper supersets can be pruned.

*Proof.* In the proof, we show that  $\text{confidence}(\mathcal{F} \rightarrow C_k) > \text{confidence}(\mathcal{F}a \rightarrow C_k)$

$$\begin{aligned} \text{Conf}(\mathcal{F} \rightarrow C_k) &= \frac{\text{wis}^e(\mathcal{F} \cup C_k)}{\text{wis}^e(\mathcal{F})} = \frac{\text{wis}^e(\mathcal{F} \cup C_k)}{\text{wis}^e(\mathcal{F}a)} \\ &> \frac{\text{wis}^e(\mathcal{F}a \cup C_k)}{\text{wis}^e(\mathcal{F}a)} = \text{Conf}(\mathcal{F}a \rightarrow C_k). \end{aligned} \quad (9)$$

□

*Pruning Rule 3.* In pruning by discrimination, given a feature  $\mathcal{F}$ , if  $\log(\sigma + \text{wis}^e(\mathcal{F} \rightarrow C_k)) / (\sigma + \text{wis}^e(\mathcal{F} \rightarrow \neg C_k)) \geq \beta$ , then  $\mathcal{F}$  will not be the discriminative prediction rules.  $\sigma$  and  $\beta$  are appointed by user.

*Proof.* If  $\log(\sigma + \text{wis}^e(\mathcal{F} \rightarrow C_k)) / (\sigma + \text{wis}^e(\mathcal{F} \rightarrow \neg C_k)) \geq \beta$ , then  $\mathcal{F}$  is relative frequent in different class. However, we say feature  $\mathcal{F}$  does not have the ability to distinguish different class because it does not satisfy Definition 4. □

The above pruning rules are very efficient since they only generate a subset of frequent features with great interestingness instead of all ones. Finally, the EC rules set is significantly smaller than an association rule set but is still too large for decision practitioners to review them all. Next, we give an ELM-based diversified feature selection method to further reduce the size of EC rules.

## 6. ELM-Based Diversified Feature Selection

As ever mentioned, the EC-Miner algorithm generates a set of optimal feature sets (rules); however, their number may be still a little large. An enormous number of features impose a great challenge on understanding and further analyzing the classification or prediction results. In this section, we study how to construct a classifier of high classification (prediction) accuracy by extracting a small number of feature sets as the representative of all mined results.

In the context of feature selection data analysis, most of the current methods adopt such a framework that ranks the attributes according to their individual discriminative power to the target class and then selects top- $k$  ranked attributes. These methods cannot remove redundant features. It is pointed out in a number of studies [22] that simply combining highly ranked features often does not form a better

feature set because these features could be highly correlated. The drawback of redundancy among selected features is twofold. On one hand, the selected feature set can have a less comprehensive representation of the target class than one of the same size but without redundant features; on the other hand, redundant features may unnecessarily increase the size of the selected feature set, which may reduce the classifier performance. Besides incapability of handling redundant features, in most ranking based methods, the number of features to be selected is arbitrarily determined.

To address the above issues, we propose an ELM-based diversified feature selection method in this section. Before describing it, we first give a diversity function as follows:

$$\begin{aligned} \text{Div}(X_1, X_2) &= \left(1 - \frac{|T(X_1) \cap T(X_2)|}{|T(X_1) \cup T(X_2)|}\right) \\ &\quad \cdot \left(1 - \frac{|\text{LCS}(X_1, X_2)|}{|I(X_1) \cup I(X_2)|}\right), \end{aligned} \quad (10)$$

where  $T(X_1)$  is the set of samples which contain  $X_1$  as a significant chain,  $I(X_1)$  is the set of items involved in  $X_1$ ,  $\text{LCS}(X_1, X_2)$  is the longest common feature of  $X_1$  and  $X_2$ , and symbol “|” denotes the length of a pattern.

In (10), the diversity between two early rules  $X_1$  and  $X_2$ , that is,  $\text{Div}(X_1, X_2)$ , is measured from two aspects: support sequences and involved items. If  $X_1$  and  $X_2$  have few common support sequences, they should have high diversity. Similarly, if  $\text{LCS}$  of  $X_1$  and  $X_2$  is short, they should have high diversity.

Based on (10), we can construct a diversity graph in the following way. For a list of results  $\text{FS} = \{X_1, X_2, \dots\}$ , the corresponding diversity graph, denoted as  $G(\text{FS}) = (V, E)$ , is an undirected graph such that, for any result  $X_i \in \text{FS}$ , there is a corresponding node  $v_i \in V$  and, for any two results  $X_i \in \text{FS}$  and  $X_j \in \text{FS}$ , there is an edge  $(v_i, v_j) \in E$  if and only if  $\text{Div}(X_i, X_j) \leq \gamma$  (a user-specified threshold). The problem of finding a set of diversified rules, which represent all mined results, is now equivalent to find an independent dominating set of  $G(\text{FS})$ . Further, we require the number of the selected features as few as possible to reduce the complexity of classifier. Thus, the problem of diversified feature selection can be viewed as an instance of finding minimum independent dominating set of  $G(\text{FS})$ , which is NP-hard [23].

Since it is difficult to find the optimal solutions, we adopt a greedy algorithm to address this problem. Given the result set  $\text{FS}$  and a set of selected results  $\text{FS}'$ , the algorithm incrementally selects patterns from  $\text{FS} - \text{FS}'$  with diversity guarantee. A pattern  $X_i \in \text{FS} - \text{FS}'$  is selected if  $\forall X_j \in \text{FS}'$ ,  $\text{Div}(X_i, X_j) > \delta$ . If there are several such alternative  $X_i$ 's in a selection, the one corresponding to a node of the most neighbors is selected. Note that, at beginning, the set  $\text{FS}'$  is empty. The algorithm picks the most significant pattern, that is, an irreducible sequence of the largest confidence value, and inserts it to  $\text{FS}'$ . As seen from what we mentioned, the final selected  $\text{FS}'$  may be more than one in the process. For example, there may be several sequences of the largest confidence value and there may be more than one node of the same number of neighbors. In such case, we use ELM

**Input:** a set of feature sets(FS)  
**Output:** The selected feature subset FS'  
(1) Let  $X$  be the feature set of the largest confidence  
(2)  $FS' = \{X\}$   
(3) **while** there is a node in  $FS - FS'$  not dominated by  $FS$  **do**  
(4) Find a pattern  $X_i \in FS - FS'$  s.t.  $\forall X_j \in FS', \text{Div}(X_i, X_j) > \delta$ ,  
and the number of the neighbors of node  $X_i$  is largest  
 $FS' = FS' \cup \{X_i\}$   
(5) **end while**  
(6) using ELM evaluates every possible FS'  
(7) the FS' of the highest accuracy on ELM;

ALGORITHM 3: The FS algorithm.

TABLE 3: QWS datasets attributes and their description.

| Attributes | # Attributes name      | Description                                                        | Units      |
|------------|------------------------|--------------------------------------------------------------------|------------|
| $X_1$      | Response time          | Time taken to send a request and receive a response                | ms         |
| $X_2$      | Availability           | Number of successful invocations/total invocations                 | %          |
| $X_3$      | Throughput             | Total number of invocations for a given period of time             | Invokes/s  |
| $X_4$      | Successability         | Number of responses/number of request messages                     | %          |
| $X_5$      | Reliability            | Ratio of the number of error messages to total messages            | %          |
| $X_6$      | Compliance             | Number of successful invocations/total invocations                 | %          |
| $X_7$      | Best practices         | The extent to which a WSDL document follows WSDL documentation     | %          |
| $X_8$      | Latency                | Time taken for the server to process a given request               | ms         |
| $X_9$      | Documentation          | Measure a documentation (i.e., description tags) in WSDL           | %          |
| $X_{10}$   | Service classification | Levels representing service offering qualities (1 through 4)       | Classifier |
| $X_{11}$   | Service name           | Name of the Web services                                           | None       |
| $X_{12}$   | WSDL address           | Location of the Web service definition language (WSDL) file on web | None       |

to evaluate every possible candidate. The one of the largest prediction accuracy is selected. Algorithm 3 formalizes the process.

The greedy algorithm can be viewed as a hybrid of the filter model and the wrapper model in feature selection, which achieves a better trade-off between the two. Better than the filter model, it explicitly removes redundancy among the selected features and determines the number of the selected features automatically. Compared with the wrapper model, it is of less computation cost.

## 7. Experiments Result Analysis

In this section, we design a series of experiments to verify the performance of the proposed method. For brevity, we refer to the algorithm of diversified feature selection based ELM as F-ELM. We select two different scenarios: one is Web service quality prediction and the other is Web service fault diagnosis prediction.

We provide two kinds of datasets. For the real dataset, we use E. AI-Mari and Dr. QH. Mahmouds' QoS dataset [20] (downloaded from <http://www.uoguelph.ca/~qmahmoud/qws/>), which includes twelve attributes ( $x_1$  to  $x_{12}$ ) as shown in

Table 3, where the attributes  $x_1$  to  $x_{10}$  are used as explanatory variables and the attribute  $x_{10}$  is used as the target variable. However, attributes  $x_{11}$  and  $x_{12}$  are ignored as they do not contribute to the analysis.

For artificial datasets, we get the Web service datasets by simulating a network environment and general network topology graph by ERITETool: with two input parameters: number of network nodes (#Web service), number of embedded classes (#class), percent of embedded fault rate (%). The system selects composite service by matching I/O operation.

*7.1. Analysis of Efficiency.* In this set of experiments, we refer to the diversified feature selection based ELM as F-ELM and the original ELM as ELM. The efficiency of F-ELM is studied by showing how response time varies with service nodes and service categories. In Figure 3, we compare the training time and the testing time between F-ELM and ELM with respect to the same categories (number of categories is 3) when the numbers of Web service are increasing (from 50 to 300). In Figures 4(a) and 4(b), we compare the training time and the testing time of F-FLM and ELM respectively, where the number of service categories varies from 2 to 8 while the number of Web services is fixed to 200.

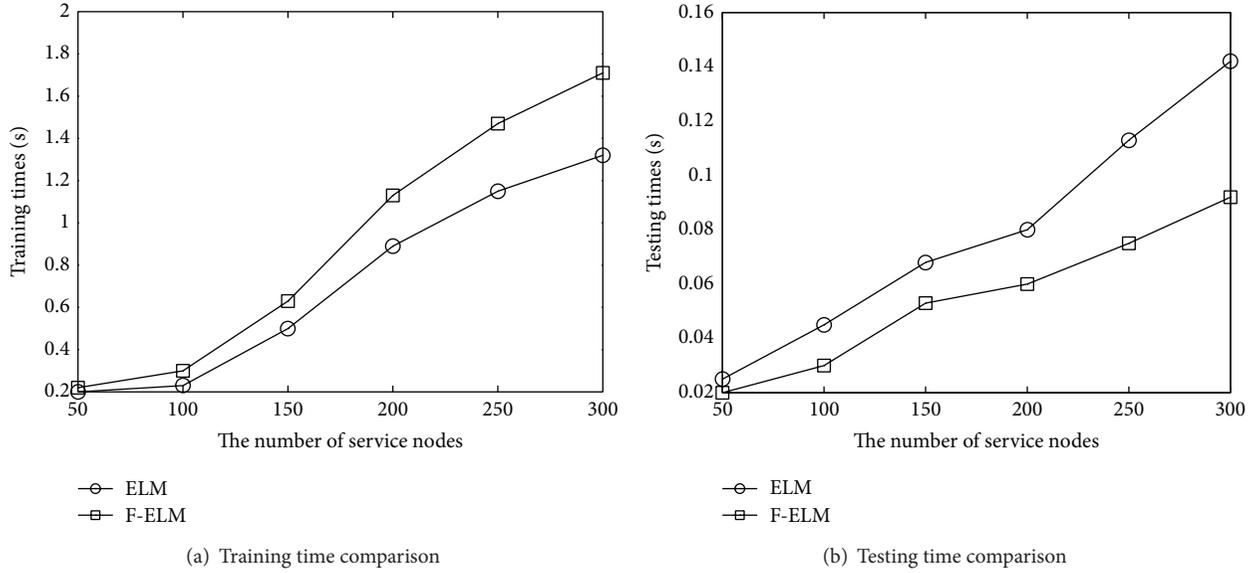


FIGURE 3: Training/testing time comparison between F-ELM and ELM.

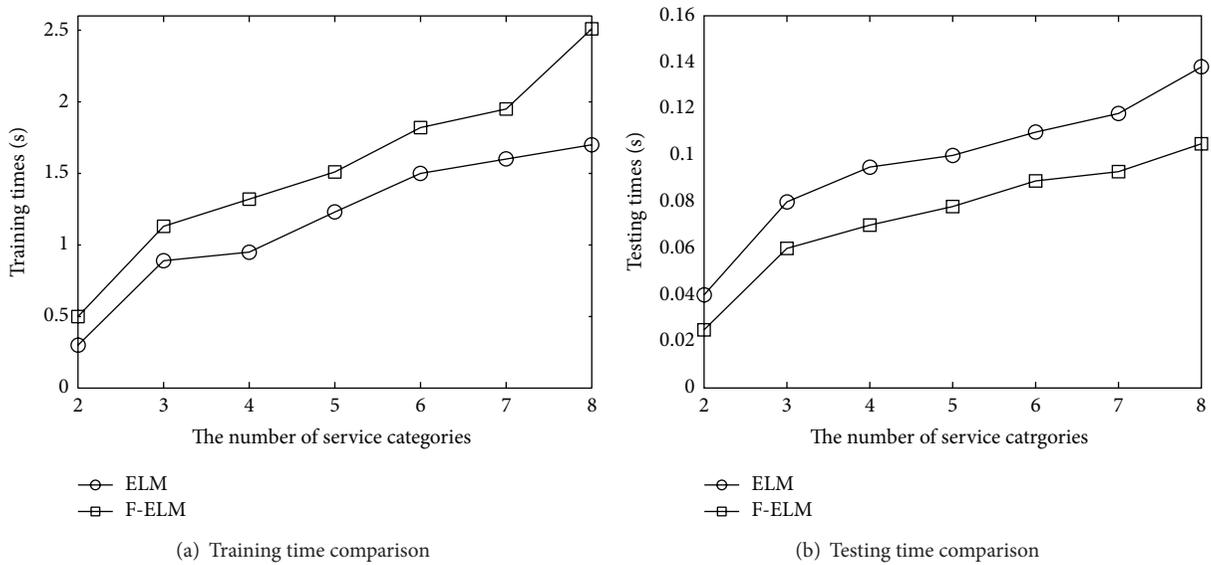


FIGURE 4: Training/testing time comparison between F-ELM and ELM.

As seen from Figures 3(a) and 4(a), training time decreases with service nodes increasing. We note that the total training time of F-ELM is a bit longer than original ELM when the service nodes are increasing. It is the same as the scenario where the service categories are increasing. This is because the increasing of service nodes (service categories) may lead to more rules to be evaluated and pruned in ELM-based diversified feature selection.

However, both Figures 3(b) and 4(b) show that the testing time of F-ELM outperforms that of ELM and the advantage becomes more substantial with a larger dataset (category). This is because ELM has to perform a time-consuming check for all feature sets. However, F-ELM only performs a series

of early and concise interesting features. Although the test time changes little, F-ELM is still constantly faster than ELM.

**7.2. Classification Accuracy.** The following evaluation criteria are used to measure the performance of F-ELM, ELM, and SVM.

Accuracy denotes the proportion of the correctly classified service sequences in the whole service sequence sets

$$\text{accuracy} = \frac{TP + TN}{N}. \tag{11}$$

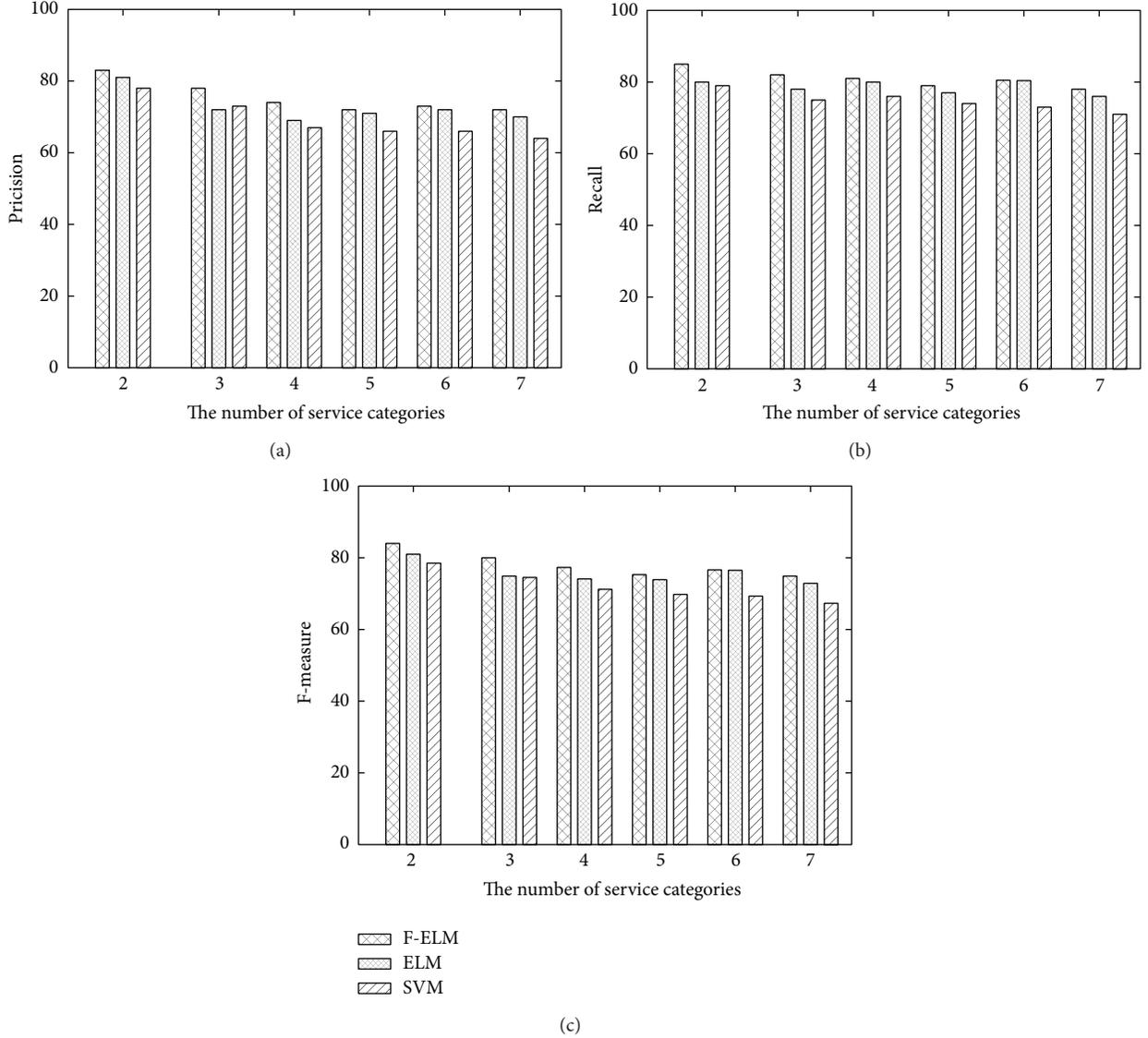


FIGURE 5: Performance comparison versus number of categories.

Precision denotes the proportion of the correctly classified service sequences with respect to a specific class

$$\text{precision} = \frac{TP}{TP + FP}. \quad (12)$$

Recall denotes the proportion of the correctly classified service sequences with respect to a specific class

$$\text{recall} = \frac{TP}{TP + FN}. \quad (13)$$

$F_1$ -measure ( $F_1$  score) is the harmonic mean of precision and recall. Since precision and recall cannot reach mathematical optimum,  $F_1$  score measures both of the two criteria and assumes that the weight of precision is equal to the weight of recall

$$F_1 = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}. \quad (14)$$

For artificial dataset, Figure 5 presents the classification comparison result of F-ELM, the original ELM algorithm, and SVM with different categories changing. The precision comparison result is presented in Figure 5(a).

The recall comparison result is presented in Figure 5(b). Figure 5(c) shows the comparison result of  $F_1$  scores. Figure 6 presents the classification comparison result of F-ELM, ELM, and SVM with different datasets changing. The precision comparison result is shown in Figure 6(a). The recall comparison result is present in Figure 6(b). Figure 6(c) shows the comparison result of  $F_1$  scores. All six figures demonstrate that F-ELM is better than ELM and SVM on each of the three criteria in terms of both the categories vary and the datasets change.

The features selected by ELM-based diversified feature selection just involve six attributes out of the original twelfth ones. To show how the selected features of the six attributes

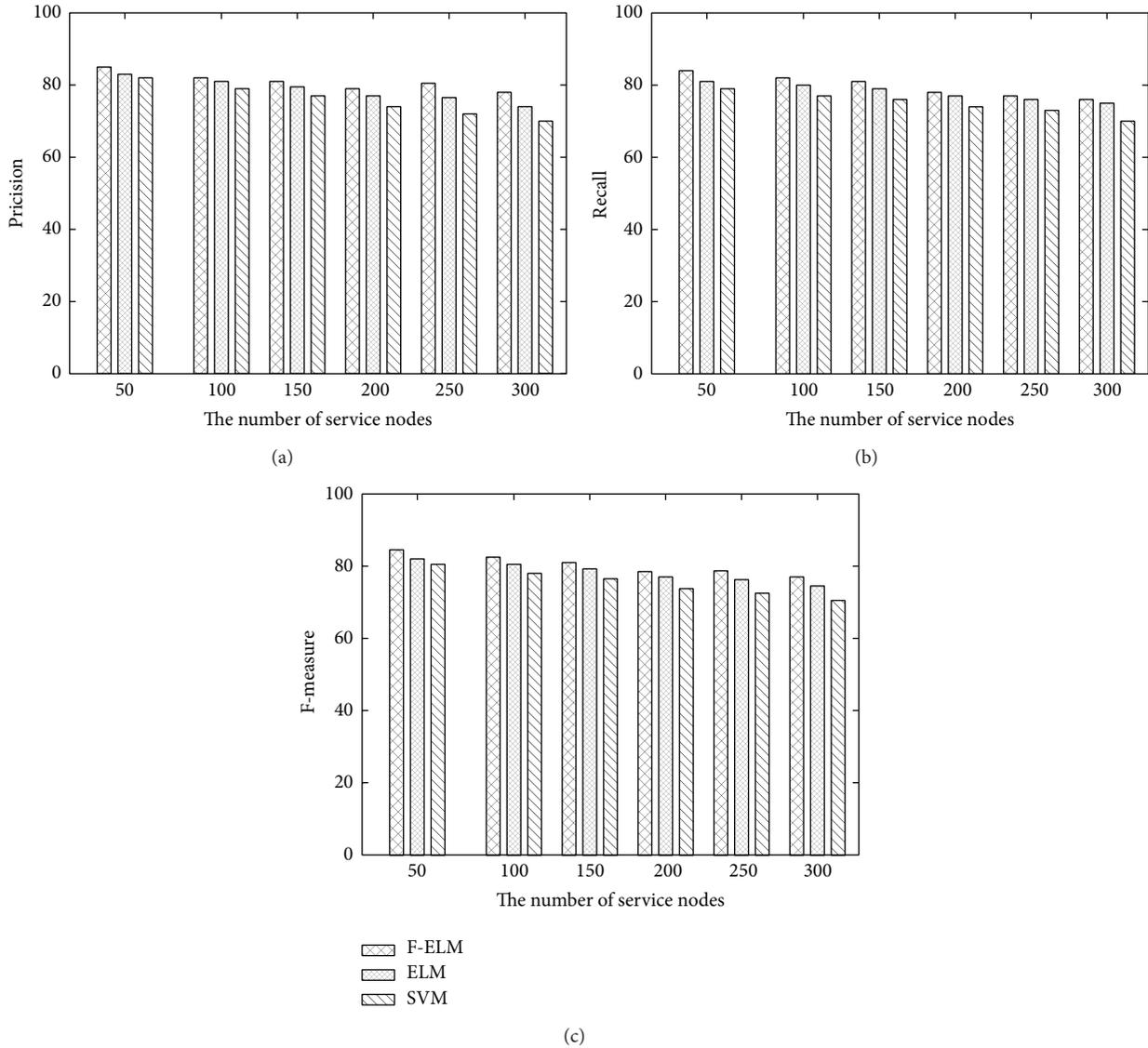


FIGURE 6: Performance comparison versus number of nodes.

affect the performance of a classifier, we compared the training time, the testing time, and the accuracies of six different classifiers, that is, ELM, SVM, CART, J48, Treenet, and BPNN, on the features of the six attributes and all the original attributes, respectively. The results are shown in Table 4. As seen, the performance of a classifier on the features of the six attributes is always better than that on the features of all the original attributes. This confirms that these classifiers can benefit from the selected features. Moreover, F-ELM behaves the best among all the introduced classifiers on the same attributes setting. This is because F-ELM exploits the relationship among attributes as the features and it removes the redundancy among the selected features while the other methods do not.  $t$ -test is utilized to evaluate whether the accuracy difference between F-ELM and a comparative method is statistically significant. Since 10-fold cross-validation is used and  $t_{0,01}(49)$  is about 2.678, the values

larger than 2.678 indicate a statistically significant difference. Thus, F-ELM does outperform the comparative methods on effectiveness. We also conduct the accuracy comparison of different algorithms on a real microarray dataset, that is, Leukemia dataset, which contains 7129 genes, 38 training samples, and 34 testing samples. The results are reported in Table 5. Since all the  $t$ -test values are larger than  $t_{0,01}(33) = 2.733$ , F-ELM still outperforms other comparative methods on accuracy in statistical significance. Thus, it is reasonable to say that the proposed method could be applied in a wider range of applications.

Additionally, we conducted a set of experiments for comparing the proposed feature selection method with six other feature selection methods, which are often used as comparative methods in machine learning for feature selection studies. The six methods are information gain (IG), twoing rule (TR), sum minority (SM), max minority (MM), Gini

TABLE 4: Accuracy comparison of different algorithms.

| Algorithm | # Attributes | Training time | Testing time | Accuracy | <i>t</i> -test |
|-----------|--------------|---------------|--------------|----------|----------------|
| F-ELM     | 6            | 1.15          | 0.26         | 82.25%   | N/A            |
| ELM       | 12           | 1.82          | 0.52         | 81.2%    | 5.103          |
| SVM       | 6            | 2.3           | 0.69         | 82.1%    | 5.211          |
| SVM       | 12           | 2.9           | 0.89         | 80.55%   | 5.060          |
| CART      | 6            | 3.1           | 0.67         | 74.1%    | 5.533          |
| CART      | 12           | 3.5           | 0.88         | 72.1%    | 5.667          |
| J48       | 6            | 3.41          | 0.96         | 66.72%   | 6.025          |
| J48       | 12           | 4.01          | 1.2          | 63.77%   | 6.222          |
| Treenet   | 6            | 2.22          | 0.74         | 77.2%    | 5.732          |
| Treenet   | 12           | 2.79          | 0.99         | 75.4%    | 5.547          |
| BPNN      | 6            | 1.87          | 0.77         | 63.1%    | 6.267          |
| BPNN      | 12           | 2.12          | 0.97         | 60.1%    | 6.467          |

TABLE 5: Accuracy comparison on Leukemia dataset.

| Algorithm | # Attributes | Training time | Testing time | Accuracy | <i>t</i> -test |
|-----------|--------------|---------------|--------------|----------|----------------|
| F-ELM     | 6            | 2.05          | 0.46         | 88.24%   | N/A            |
| ELM       | 12           | 3.26          | 0.93         | 85.29%   | 7.033          |
| SVM       | 6            | 2.71          | 0.81         | 82.35%   | 7.194          |
| SVM       | 12           | 3.42          | 1.05         | 82.35%   | 7.194          |
| CART      | 6            | 4.37          | 0.94         | 76.47 %  | 7.780          |
| CART      | 12           | 4.94          | 1.24         | 73.53%   | 7.989          |
| J48       | 6            | 4.71          | 1.33         | 67.65%   | 8.493          |
| J48       | 12           | 5.54          | 1.66         | 61.76%   | 8.771          |
| Treenet   | 6            | 3.65          | 1.22         | 76.47%   | 7.780          |
| Treenet   | 12           | 4.23          | 1.57         | 73.53%   | 7.876          |
| BPNN      | 6            | 3.13          | 1.29         | 64.71%   | 8.643          |
| BPNN      | 12           | 3.56          | 1.63         | 61.76%   | 9.116          |

TABLE 6: Accuracy comparison versus different feature selection algorithms.

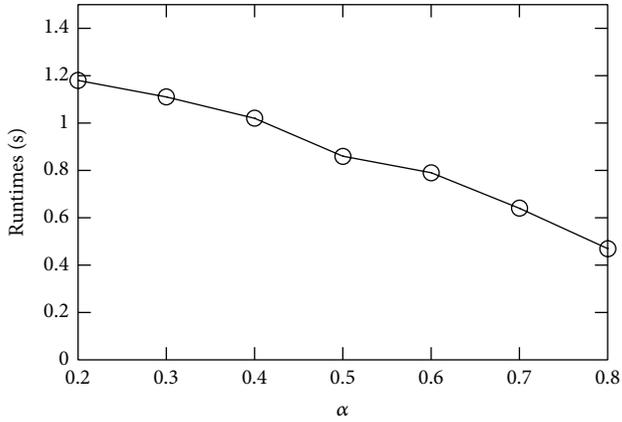
| Algorithm | EF     | IG     | TR     | SM     | MM     | GI     | SV     |
|-----------|--------|--------|--------|--------|--------|--------|--------|
| ELM       | 82.55% | 81.47% | 79.83% | 77.66% | 78.08% | 81.72% | 82.3%  |
| SVM       | 82.1%  | 80.64% | 79.0%  | 78.6%  | 77.1%  | 79.65% | 81.03% |
| CART      | 74.1%  | 71.03% | 71.75% | 72.01% | 73.11% | 71.68% | 71.68% |
| J48       | 66.72% | 64.37% | 59.76% | 55.51% | 58.41% | 61.79% | 59.63% |
| Treenet   | 77.2%  | 73.8%  | 77.7%  | 74.48% | 74.73% | 73.68% | 74.67% |
| BPNN      | 63.1%  | 60.63% | 58.8%  | 58.5%  | 59.07% | 59.34% | 60.73% |

index (GI), and sum of variance (SV), respectively. The results are reported in Table 6, where ELM-based feature selection is abbreviated as EF. As seen from Table 6, EF approach provides the highest accuracy on all classifiers. This is mainly because the features selected by EF can be considered as a diversified coverage of all the original features, which provide more complete but less redundant information than the comparative methods.

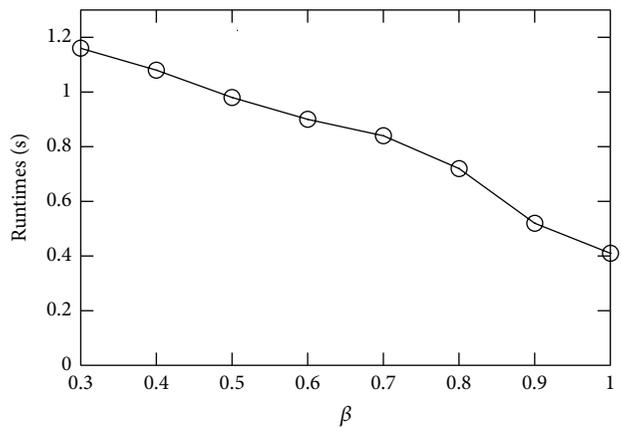
In summary, as seen from Tables 4 and 5, when applying ELM in service QoS prediction or in other applications (such as microarray data classification), we can always obtain the

best results. This confirms the effectiveness of ELM in a wide of applications. Further, this can be explained in such a way that the proposed F-ELM extracts the concise features of high discriminative power for each category while the other methods do not.

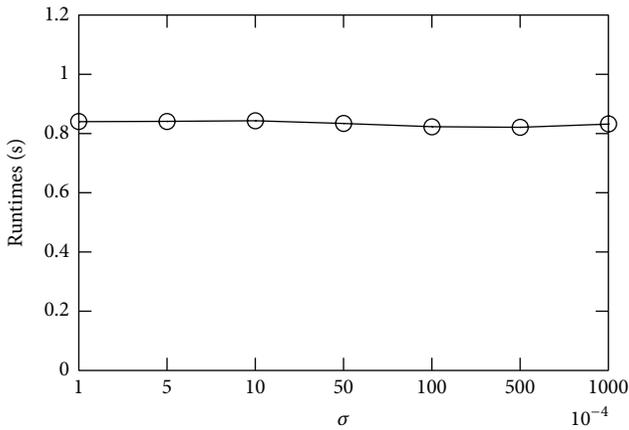
*7.3. Effects of the Parameters.* In this section, we study how the parameters affect the performance of ELM-based diversified feature selection. In Figure 7(a), the running time of the feature selection decreases with  $\alpha$  increasing. This is because the larger  $\alpha$  makes more rules pruned. The reduced search



(a)



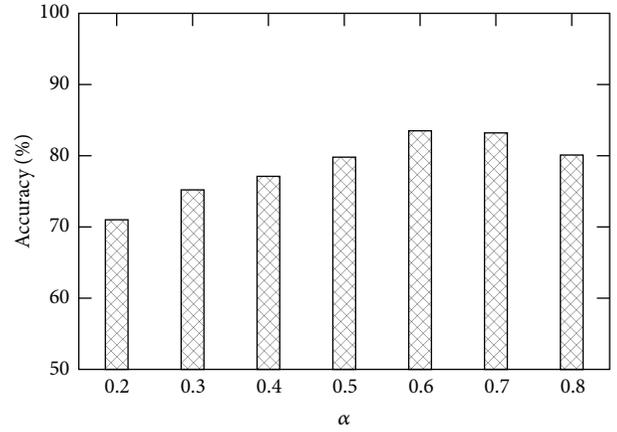
(b)



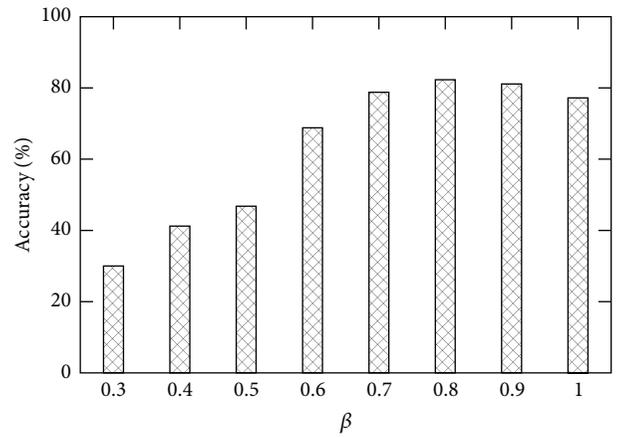
—○— F-ELM

(c)

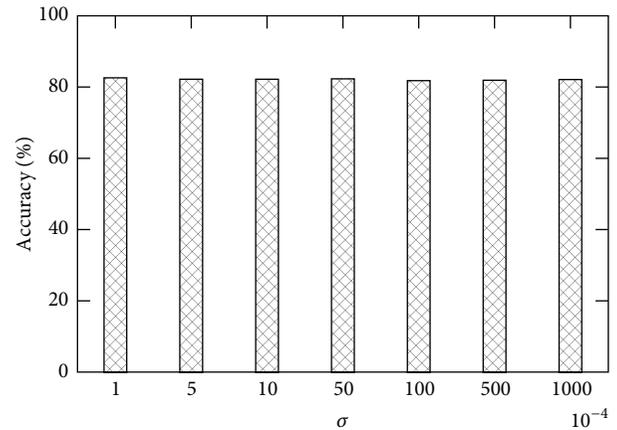
FIGURE 7: Runtime versus parameters.



(a)



(b)



—▨— F-ELM

(c)

FIGURE 8: Accuracy versus parameters.

space leads to the less running time. However, this does not indicate that we should choose  $\alpha$  as large as possible. Figure 8(a) shows that too large  $\alpha$  may deteriorate classification accuracy. This is because many rules of potentially high usability will be pruned at a high  $\alpha$  level. Also, the accuracy at a too low  $\alpha$  level is not very good due to the “overfitting”

problem. Figures 7(b) and 8(b) give how  $\beta$  affects the feature selection performance. The cases are similar as those in Figures 7(a) and 8(a), respectively. That is, the running time of the feature selection decreases with  $\beta$  increasing, and too large and too low  $\beta$  may deteriorate classification accuracy. The results can also be explained in a similar way as those

for Figures 7(a) and 8(a), respectively. Differently, Figures 7(c) and 8(c) show that  $\sigma$  rarely affects the running time of the feature selection and classification accuracy. This is because  $\sigma$  is introduced just for avoiding the case where the denominator of (8) is zero.  $\sigma$  is often set to a very low value, the effect of which is dominated by other values setting in (8).

## 8. Conclusions

In this paper, we propose an ELM-based service quality prediction framework. Considering the highly dynamic and the uncontrollable circumstances, the service quality prediction is required to be triggered as soon as possible in the proposed framework. By developing the prefix tree based algorithm, EC-Miner, a series of candidate rule sets are first found, where both the earliness and conciseness of the rules are considered. Then, an ELM-based diversified feature selection algorithm is proposed to fine the candidate rule set. A small subset of high-quality features are discovered as the representative of the whole candidate rule set. A greedy algorithm is presented to approximate the optimal solution. Experimental results show that the proposed approach significantly improves the efficiency and the effectiveness of ELM with respect to some widely used feature selection techniques.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China under Grants nos. 61100028, 61272182, 61073062, and 61073063; State Key Program of National Natural Science of China (61332014); the New Century Excellent Talents in University Award (NCET-11-0085); the Fundamental Research Funds for the Central Universities under grants (no. 130504001); the Ph.D. Programs Foundation of Ministry of Education of China (young teacher) (no. 20110042120034).

## References

- [1] L. Zhang, J. Zhang, and C. Hong, *Services Computing*, Tsinghua University Press, Beijing, China, 2007.
- [2] F. Wang, L. Liu, and C. Dou, "Stock market volatility prediction: a service-oriented multi-kernel learning approach," in *Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12)*, pp. 49–56, June 2012.
- [3] J. W. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Machine Learning Press, 3rd edition, 2012.
- [4] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, Hangzhou, China, June 2014.
- [5] A. Goldman and Y. Ngoko, "On graph reduction for QoS prediction of very large web service compositions," in *Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12)*, pp. 258–265, June 2012.
- [6] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 573–579, 2013.
- [7] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service QoS prediction with location-based regularization," in *Proceedings of the IEEE 19th International Conference on Web Services (ICWS '12)*, pp. 464–471, Honolulu, Hawaii, USA, June 2012.
- [8] J. Wu, L. Chen, H. Jian, and Z. Wu, "Composite service recommendation based on bayes theorem," *International Journal of Web Services Research*, vol. 9, no. 2, pp. 69–93, 2012.
- [9] J. Park, H. Yu, K. Chung, and E. Lee, "Markov chain based monitoring service for fault tolerance in mobile cloud computing," in *Proceedings of the 25th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '11)*, pp. 520–525, March 2011.
- [10] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [11] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [12] X.-G. Zhao, G. Wang, X. Bi, P. Gong, and Y. Zhao, "XML document classification based on ELM," *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, 2011.
- [13] G. Wang, Y. Zhao, and D. Wang, "A protein secondary structure prediction framework based on the Extreme Learning Machine," *Neurocomputing*, vol. 72, no. 1–3, pp. 262–268, 2008.
- [14] J. Cao and Z. Lin, "Bayesian signal detection with compressed measurements," *Information Sciences*, vol. 289, pp. 241–253, 2014.
- [15] R. Zhang, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Multi-category classification using an Extreme Learning Machine for microarray gene expression cancer diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 3, pp. 485–495, 2007.
- [16] Y. Zhao, J. Y. Xu, G. Wang, L. Chen, B. Wang, and G. Yu, "Maximal subspace co-regulated gene clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 83–98, 2008.
- [17] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.
- [18] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, no. 1–4, pp. 306–314, 2005.
- [19] Y. Zhao, G. Wang, X. Zhang, J. X. Yu, and Z. Wang, "Learning phenotype structure using sequence model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 667–681, 2014.
- [20] R. Mohanty, V. Ravi, and M. R. Patra, "Web-services classification using intelligent techniques," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5484–5490, 2010.
- [21] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Addison-Wesley, 2nd edition, 2014.
- [22] J. Biesiada and W. Duch, "Feature selection for high-dimensional data—a person redundancy based filter," in

*Computer Reognition System*, Advances in Soft Computing, pp. 242–249, Springer, Berlin, Germany, 2008.

- [23] D. Zuckerman, “On unapproximable versions of NP-complete problems,” *SIAM Journal on Computing*, vol. 25, no. 6, pp. 1293–1304, 1996.

## Research Article

# Extreme Learning Machine for Reservoir Parameter Estimation in Heterogeneous Sandstone Reservoir

Jianhua Cao, Jucheng Yang, Yan Wang, Dan Wang, and Yancui Shi

*College of Computer Science and Information Engineering, Tianjin University of Science and Technology, Tianjin 300222, China*

Correspondence should be addressed to Jucheng Yang; [jcyang@tust.edu.cn](mailto:jcyang@tust.edu.cn)

Received 21 August 2014; Revised 8 November 2014; Accepted 10 November 2014

Academic Editor: Zhan-li Sun

Copyright © 2015 Jianhua Cao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This study focuses on reservoir parameter estimation using extreme learning machine in heterogeneous sandstone reservoir. The specific aim of work is to obtain accurate porosity and permeability which has proven to be difficult by conventional petrophysical methods in wells without core data. 4950 samples from 8 wells with core data have been used to train and validate the neural network, and robust ELM algorithm provides fast and accurate prediction results, which is also testified by comparison with BP (back propagation) network and SVM (support vector machine) approaches. The network model is then applied to estimate porosity and permeability for the remaining wells. The predicted attributes match well with the oil test conclusions. Based on the estimations, reservoir porosity and permeability have been mapped and analyzed. Two favorable zones have been suggested for further research in the survey.

## 1. Introduction

In geosciences, reservoir is defined as the underground accumulation of oil or natural gas in sedimentary basins, and it is of great importance for petroleum exploration and development. Among the steps for well planning decisions, reservoir characterization is the essential one, and physical parameters estimation, including porosity and permeability, is the basic requirement in the characterization workflow.

As for the two geophysical parameters, porosity describes the fraction of void space in the sedimentary rocks, where the void may contain fluids, such as oil or natural gas. The more porous the rock is, the more the oil or gas may be preserved in the void spaces. And permeability describes the ability of rocks to transmit fluids. The more permeable the rock is, the easier the oil or gas could flow through. These two types of reservoir parameters are to some extent determining factors for reserve estimation and oil or gas production.

Practically, it is very complex and difficult for porosity and permeability estimation since lots of factors could affect the estimation accuracy, such as depositional formations, lithologic mineral components, measurement tools, data quality, and computational method.

There are mainly two types of approaches that have been used to acquire porosity and permeability data in reservoir research workflow. The first one is laboratory core analysis. Cores are obtained from drilled wells. Porosity and permeability can be determined precisely under strict core test principles. The results are reliable and are often used as reference for further estimation using mathematical ways. Due to the expensive cost, cores are often few in numbers for most of the oilfields. The second one is borehole log interpretation. The logs data are physical measurements performed by electric instruments lowered into the borehole. Specific physical characteristics of the rocks surrounding the borehole are recorded by logs with depth variations. Conventional logs include gamma ray (GR), acoustic slowness (AC), density (DEN), compensated neutron logs (CNL), and deep resistivity (RT). Among these logs, GR is often used to predict rock lithology, and the three logs of AC, DEN, and CNL have largely been used to estimate rock porosity. Permeability is estimated by combination of RT log and the former-estimated porosity. Empirical mathematical equations are often used when carrying out log interpretations. These equations are regression models built based on the correlation between geophysical logs and core-measured reservoir parameters. Since logs are

run for all wells in oilfield and the mathematical empirical equations are feasible to be used, log interpretation becomes the most used method in porosity and permeability estimation. But the estimation results rely greatly on the equations or the correlation models. Meanwhile, the relations between logs and geophysical parameters of rocks are nonlinear and very complicated. It is hard to get a universal solution for all wells in one survey or for all oilfields. So some nonlinear numerical method and artificial intelligence are brought into the log interpretation process and proposed as supplementary approach, so that more reliable and precise estimation data could be obtained for further reservoir evaluation.

Artificial neural networks have been proved to be capable of approximating any nonlinear function to any degree of accuracy provided that there are sufficient number of samples for network training and learning and have some successful applications in petroleum engineering, such as sedimentary microfacies prediction [1], lithology classification [2, 3], and reservoir prediction [4–6].

In petrophysical analysis, the neural network models have always acted as a predictor or estimator of deriving geophysical parameters, such as porosity and permeability where no core data is available [7–12]. Among the neural networks, BP network and SVM are the two commonly used learning algorithms in porosity and permeability estimation. BP neural network is a typical full-connected neural network with forward and error backpropagation part. The error could be backpropagated by adjusting weights in the learning process until it converges to a targeted value, which is very effective in solving nonlinear problems [13]. Support vector machine (SVM) is a network based on statistical learning theory and is especially designed for classification problems with different convolution kernel functions [14]. Satisfactory accuracy of estimations has been achieved when the networks are optimized with appropriate model parameters [15–19], although there still are some shortcomings in the applications, such as time-consuming and overfitting problems [20].

Extreme learning machine (ELM) is a single-hidden layer feedforward neural network (SLFN) proposed by Huang et al. [21, 22]. The ELM approach to training SLFN consists in the random generation of the hidden layer weights, followed by solving a linear system of equations by least squares for the estimation of the output layer weights. This learning strategy is very fast and gives good prediction accuracy. Theoretically and practically, this algorithm can produce good generalization performance in most cases and the speed has been proved to be much faster than conventional popular learning algorithms for feedforward neural networks. Till now, ELM has been widely studied and accepted by researchers and has demonstrated good generalization and prediction performance in many real-life applications [23–26]. But in petroleum reservoir prediction, there are still few applications.

In this paper, we examine the potential of ELM to predict porosity and permeability parameters in a heterogeneous sandstone reservoir in Permian formation, Yanqi survey of Ordos basin, China. Prediction models are established by SLFN trained with ELM and optimally pruned ELM (OP-ELM). OP-ELM is a variation of the ELM introducing

an optimal selection of the number of hidden units and variables modeling the problem [27, 28]. It is more robust and generic than conventional ELM [26]. In the study, reservoir parameters measured from cores and logs value at the same depth are paired as samples. Optimal prediction models for porosity and permeability estimation are established, which are finally used to interpret reservoir porosity and permeability for all wells in the survey.

The outline of this paper is as follows: Section 2 is the geologic background of the survey and brief introduction about the sandstone reservoir. Section 3 gives a short review of ELM and OP-ELM. Section 4 describes the preparation for the network model establishment. Section 5 gives the prediction results. Finally, Section 6 gives the conclusion of this work.

## 2. Geological Background

Yanqi survey is located in eastern Ordos basin, China. In the survey, there are 15 wells that encountered Permian sandstone reservoir. Oil has been discovered in 6 wells, which are marked with red color-filled circles in Figure 1.

According to the comprehensive study of Permian formation, the structure is monoclinical and west-dipping with few faults developed. The Permian formation is dominated by calcareous quartz sandstone, interbedded with organic-rich mudstone and thin-layer coal.

Owing to the complex depositional process and variable diagenesis, the sandstone reservoir is heterogeneous and changes fast spatially, which is controlled by the fluvial sedimentary microfacies. Core analysis shows that the main pore type is intragranular pore (Figure 2(b)), with proportion more than 55%, and the rest include secondary intergranular pore (Figure 2(c)), matrix pore, and microcrack. And the measured permeability varies greatly both horizontally and vertically.

Figure 3 is the cross-plot of actual porosity and permeability measured from core samples. It shows that porosity ranges from 2% to 14% and permeability value has large span from 0.0001 md to 10 md. The two parameters have highly positive correlation. Exponential regression has been executed for porosity and permeability, which is

$$\text{Permeability} = 0.0009e^{0.6567\text{Porosity}} \quad (1)$$

and the correlation coefficient ( $R$ ) is 77.9%. In conventional log interpretation, this statistical regression model can be recommended for permeability estimation if porosity is determined, but practically porosity is hard to be accurately calculated using current empirical formula. What is more, the correlation coefficient is not good enough, and the error of porosity calculation can also be brought into the permeability estimation process and affect its accuracy. So the model may not be suitable for all wells of the survey.

According to the petroleum industry criteria, the sandstone reservoir in Yanqi survey has medium porosity and permeability. In the survey, 6% is set as the threshold of porosity for sandstone reservoir, which means if porosity is lower than

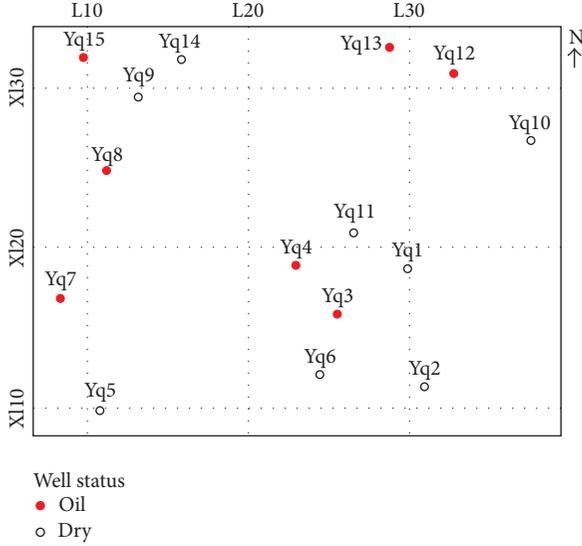


FIGURE 1: Well location and base map of the Yanqi 3D survey. There are totally 15 wells, with 7 wells encountering oil in Permian sandstone reservoir, while the rest are dry.

the threshold, the sandstone reservoir is considered tight and uneconomic.

### 3. Extreme Learning Machine

In this section we review the fundamental definitions of the two learning algorithms applied in the experiments reported below to predict porosity and permeability parameters in Permian sandstone: the ELM and the OP-ELM.

**3.1. ELM.** Extreme Learning Machine (ELM) is a simple supervised learning algorithm proposed by Huang et al. [21] for Single-hidden Layer Feedforward Neural Network (SLFN). Comparing with the conventional neural networks, ELM has better performance in learning efficiency and universal approximation capability. Different from BP network, the input weights and biases of ELM are randomly assigned and need not be adjusted within the training phase, and the output weights can be determined analytically by finding the least squares solution. Therefore the neural network is obtained after a few steps with very low computation cost.

Given a dataset containing  $N$  training samples  $(\mathbf{x}_i, \mathbf{t}_i)$ ,  $i = 1, 2, \dots, N$ , where  $\mathbf{x}_i$  is a  $n \times 1$  input vector  $\mathbf{x}_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^n]^T \in \mathbf{R}^n$  and  $\mathbf{t}_i$  is a  $m \times 1$  target vector  $\mathbf{t}_i = [\mathbf{t}_i^1, \mathbf{t}_i^2, \dots, \mathbf{t}_i^m]^T \in \mathbf{R}^m$ , the output of a SLFN with  $M$  hidden nodes can be expressed mathematically as

$$\sum_{i=1}^M \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^M \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + \mathbf{b}_i) = \mathbf{t}_j, \quad (2)$$

$$j = 1, 2, \dots, N,$$

where  $\mathbf{w}_i$  is the weight vector connecting the  $i$ th hidden node with the input nodes,  $\beta_i$  is the weight vector connecting the  $i$ th hidden node with the output nodes, and  $\mathbf{b}_i$  is the threshold

of the  $i$ th hidden node.  $g(\cdot)$  denotes the nonlinear activation function of the hidden node. It can be the identity sigmoid or Gaussian function, among a large collection of polynomial functions.

Equation (2) can be written in a more compact format as follows:

$$\mathbf{H}\beta = \mathbf{T}, \quad (3)$$

where  $\mathbf{H}$  is the hidden layer output matrix of the network:

$$\mathbf{H} = \begin{bmatrix} g(w_1 x_1 + b_1) & \dots & g(w_N x_1 + b_N) \\ \vdots & \ddots & \vdots \\ g(w_1 x_M + b_1) & \dots & g(w_N x_M + b_N) \end{bmatrix}, \quad (4)$$

$\beta$  is the matrix of hidden-to-output weights, and  $\mathbf{T}$  is the target matrix.

In (4), weights ( $\mathbf{w}_i$ ) and biases ( $\mathbf{b}_i$ ) are randomly assigned and  $g(\cdot)$  is known to be selected as sigmoid function, so the output of hidden nodes could be determined, which is  $\mathbf{H}$  in (3). The remaining problem becomes a set of linear equations and can be solved by minimum square error estimation:

$$\text{Min}_{\beta} \|\mathbf{H}\beta - \mathbf{T}\|. \quad (5)$$

According to the definition of the Moore-Penrose generalized inverse, the smallest norm least squares solution of (3) is given as

$$\hat{\beta} = \mathbf{H}^{-1}\mathbf{T}, \quad (6)$$

where  $\mathbf{H}^{-1}$  is the Moore-Penrose generalized inverse of matrix  $\mathbf{H}$ .

For (6), once the  $\mathbf{H}$  and  $\mathbf{T}$  are set, it is not difficult to get the  $\beta$  matrix. The process has proved several advantages: (1) the training error is minimized; (2) the generalization performance is optimal; (3) the solution is unique.

Totally, the ELM algorithm can be summarized as follows [24].

Given a training set  $\{(x_i, t_i) \mid x_i \in \mathbf{R}^n, t_i \in \mathbf{R}^m, i = 1, \dots, N\}$ , an activation function  $g$ , and the number of hidden nodes  $M$ ,

- (1) randomly set input weights  $\mathbf{w}_j$  and biases  $\mathbf{b}_j$ ;
- (2) calculate the hidden layer output matrix  $\mathbf{H}$ ;
- (3) calculate the output weight matrix  $\hat{\beta}$ .

**3.2. OP-ELM.** The optimally pruned extreme learning machine (OP-ELM) is a variation of ELM algorithm for SLFN. The OP-ELM algorithm is made of three main steps summarized as follows [27, 28].

Given a training set  $\{(x_i, t_i) \mid x_i \in \mathbf{R}^n, t_i \in \mathbf{R}^m, i = 1, \dots, N\}$ .

(1) Build a regular ELM model with initially large number of neurons.

(2) Rank the hidden layer neurons by their contribution to the linear explanation of the ELM output by the multiresponse sparse regression (MRSR), which was proposed by Similä and Tikka in 2005 [29].

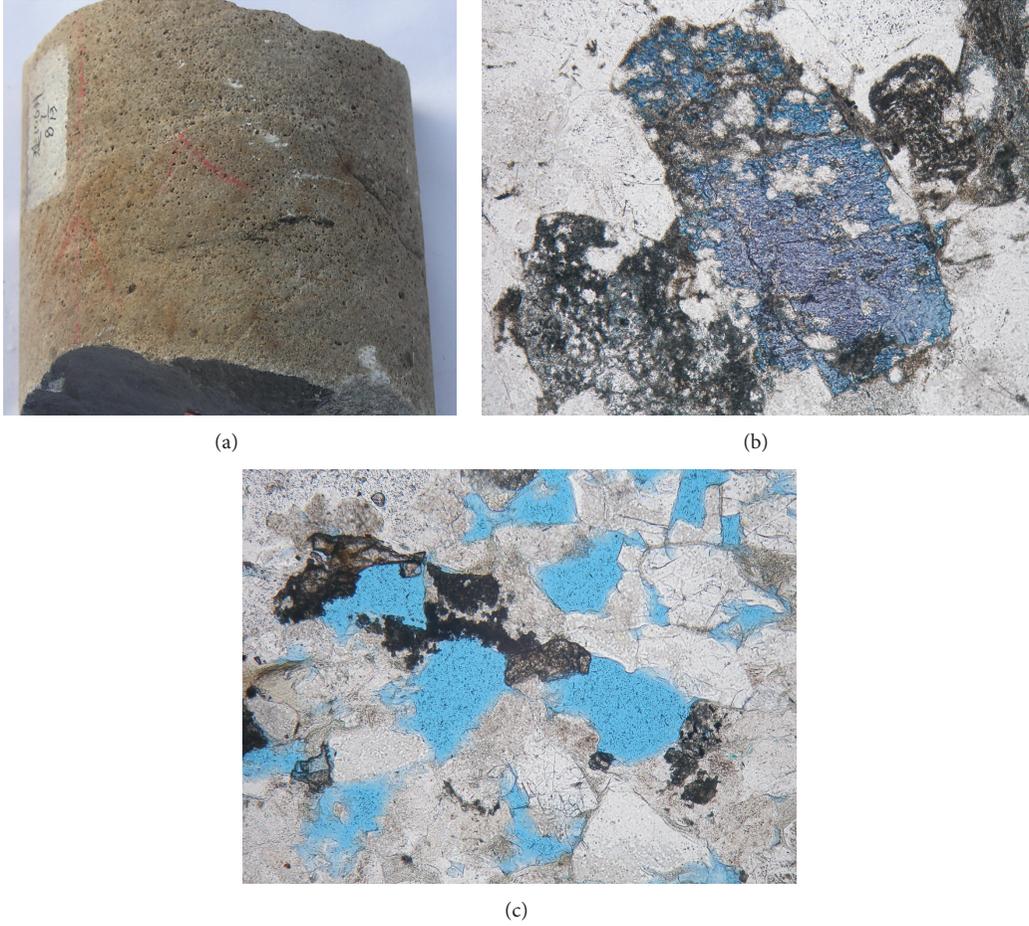


FIGURE 2: Photo and microscope slice of core samples from Permian formation of well Yq3 in Yanqi survey: (a) actual core photo, (b) intragranular pore, and (c) intergranular pore. Small pores can be recognized on the side of core. In the microscope slices, pore spaces are colored and the grey part is the matrix.

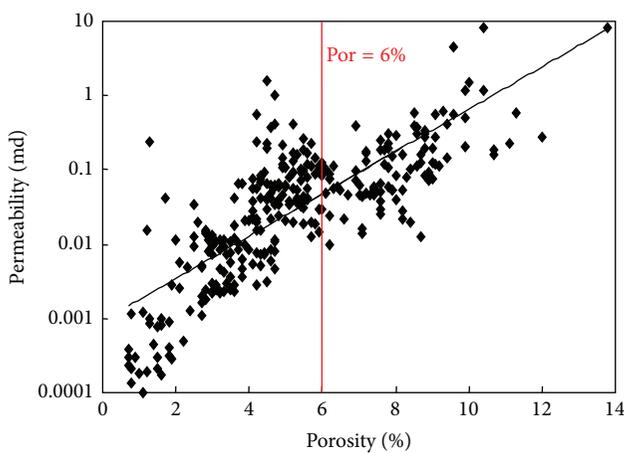


FIGURE 3: Cross-plot of core-measured porosity and permeability.

MRSR is used to get rid of the useless neurons of the hidden layer. Its main idea is as follows: add columns of the regressor matrix  $\mathbf{H}$  in  $\mathbf{H}\hat{\beta} = \mathbf{T}$  and corresponding nonzero

rows in  $\hat{\beta}$  and thus obtain a series of approximations  $\mathbf{H}^k\hat{\beta} = \mathbf{T}^k$ . Then hidden nodes are ordered by the corresponding decrease in the prediction error  $\|\mathbf{T}^k - \mathbf{T}\|$  obtained in the model. More specific details of the MRSR algorithm can be found in [29].

(3) Decide the optimal number of neurons by the leave-one-out (LOO) validation method. Compute the LOO using the PRESS (prediction sum of squares) statistic in the linear case:

$$\epsilon^{\text{PRESS}} = \frac{t_i - \mathbf{h}_i \mathbf{b}_i}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T}, \quad (7)$$

where  $\mathbf{h}_i$  and  $\mathbf{b}_i$  are the  $i$ th column and  $i$ th row of  $\mathbf{H}$  and  $\hat{\beta}$ , respectively. The process is greedily incremental, and units are added in order until the LOO method decreases below a preset threshold.

For robustness and more generality, the OP-ELM algorithm has the suggestion of using a combination of three types of kernels: linear, sigmoid, and Gaussian kernel, while the original ELM proposed to use only sigmoid kernels [21]. Problems discussed in this paper are not linear, and

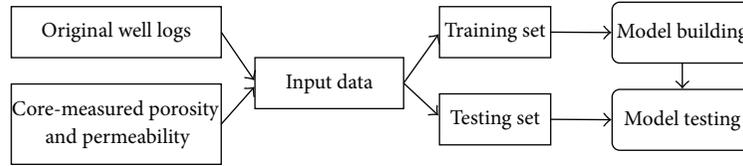


FIGURE 4: Experimental flow diagram of the estimation experiments.

the experiments conducted below will compare the efficiency using different kernel types and choose the optimal one for both ELM and OP-ELM.

#### 4. Experimental Design

The experimental analysis is conducted to find the best model for predicting porosity and permeability from log data. Figure 4 shows the flow diagram of the computational experiments carried out. Rectangle boxes correspond to data, while rounded boxes correspond to process. For the experiment, the first step is to prepare the input data, including the original log data and core-measured porosity and permeability data. The datasets are divided into training set and testing set. The second step is to set up the network model with the training dataset. Model parameters, including kernel types and neuron numbers of hidden layers, are determined. The third step is to validate the model using the testing dataset. If the error is below the threshold, the model is feasibly and appropriately built; thereafter it can be used in reservoir parameters prediction for the remaining wells.

**4.1. Data Preparation.** In the survey, all of the wells in the survey have run the conventional logging, and the five types of logs for reservoir parameters estimation are well prepared. About 8 wells have cores from Permian sandstone formation. Most of the cores have laboratory analysis results, including the measured porosity and permeability values. Unlike the continuously recorded well logs data, the core data are sparsely sampled (as shown in Figure 5). So when collecting input data for the network, each core-measured data matches with the log data at the same depth.

Five types of logs including AC, DEN, GR, CNL, and RT are used as input for the networks, and porosity (POR) and permeability (PERM) measured from cores are the two output targets. Two parts are prepared using the datasets, including training part and test part. The training part is used to train the model, while the test part is used to compute predictions and compare them with the measured values. Mean square error (MSE) is computed as evaluation of the prediction accuracy and quality of the trained model.

The total number of samples in Permian sandstone of 8 wells of the survey is summed up to 4950, 90% of which will be used as the training data for the ELM network, while the remaining 10% will be used as test samples.

Data normalization is a necessary preprocessing step for network data analysis. So all of the logs and core-measured

parameters are normalized before formally inputting into the network. The normalized variable has the following form:

$$X_{\text{new}} = \frac{X_{\text{old}} - \min X}{\max X - \min X}, \quad (8)$$

where  $X$  stands for logs of GR, AC, DEN, CNL, and RT. The new normalized variable  $X_{\text{new}}$  takes the range from 0 to 1 for all the parameters.

**4.2. Network Architecture.** For porosity and permeability estimation, five logs including GR, AC, DEN, CNL, and RT are physically related to petrophysical properties. So these five logs are fed to the ELM network as input with each node denoting one log. Porosity and permeability are to be taken as the two network neurons at the output layer. The network architecture is shown as Figure 6.

**4.3. Network Parameter Selection.** For ELM network, appropriate kernel and number of hidden nodes are the two critical parameters to be determined.

In the study, four types of kernels have been tested using training dataset, and they are sigmoid function, radial basis function, hardlim function, and triangular function. At the same time, numbers of hidden nodes are tested accordingly. When selecting one of the kernels, number of hidden nodes will start from 5, with 10 as incremental step. Figure 7 is the accuracy comparison by using different kernels and node numbers. The MSE between prediction results and measured properties decreases rapidly as node number of the hidden layer gradually increases. Among the four kernels, sigmoid-based model comes to the threshold first when node number is set as 55, while overfitting problem appears as the node number is bigger than 65. Triangular-based and radial-basis-based models have the same trend, and it seems that the node number might exceed 100 when the minimum errors are close to the threshold. For the hardlim-based model, the MSE reaches the lowest point at 3.26% when node number is set as 75.

Based on the experimental tests, sigmoid kernel is optimal and when node number of the hidden layer is set as 55, the network model can obtain the best prediction accuracy.

For the ELM network model, node number of hidden layer can also be optimally determined by OP-ELM training process instead of time-consuming and arbitrary testing. The very first step of OP-ELM methodology is to construct the SLFN network using the original ELM structure with 100 neurons at the hidden layer and sigmoid kernel. Both of training dataset and testing dataset are input to the model,

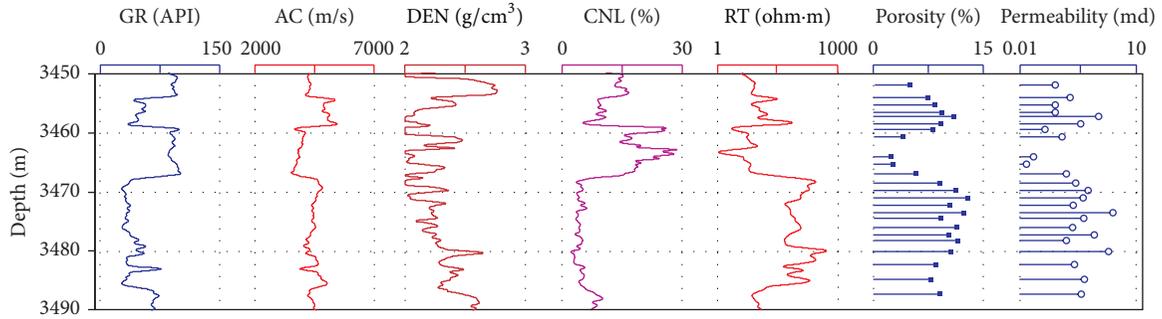


FIGURE 5: Part of logs and corresponding core-measured porosity and permeability in Permian formation of well Yq12. The measured parameters are sparsely distributed along the borehole, while logs are continuously sampled, and the sampling interval of logs is 0.5 m.

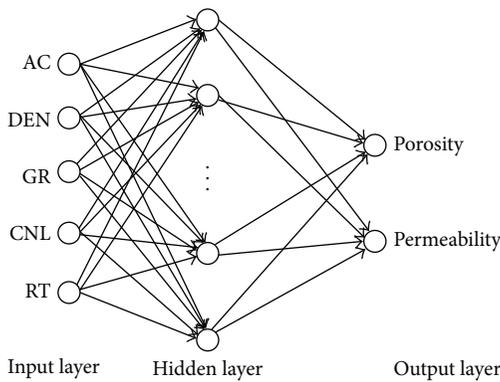


FIGURE 6: Schematic of architecture of the ELM network model. Node number of the input layer is 5, and porosity and permeability are the two nodes at the output layer. Number of hidden layer nodes is to be settled by OP-ELM method.

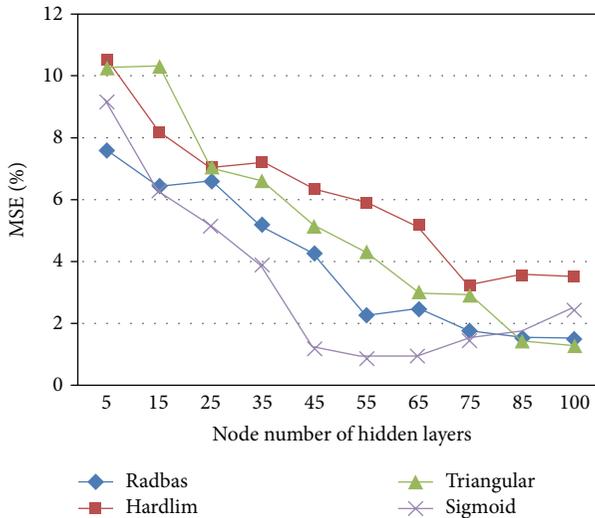


FIGURE 7: Comparison of mean squared error (MSE) obtained from different kernel-based ELM models with increasing node numbers at the hidden layers.

and MSEs are computed for looking for the best accuracy. Throughneuron contribution sorted by MRSR and optimal

nodes selected by LOO validation, the optimal neuron at the hidden layer is finally determined as 62, which is close to the original ELM test.

4.4. Accuracy Analysis. When the parameters for the network model are finally settled, the following step is to validate the model using testing dataset. Core porosity and permeability are set as targets to the network predictor. Figure 8 is the validation result for the well Yq12 with depth ranging from 3450 m to 3490 m (mentioned in Figure 5). The model outputs are superimposed on the core data. Regression plots in Figures 9(a) and 9(b) reflect the accuracy of the OP-ELM network estimator. Coefficients of 0.9932 and 0.9917 are obtained for porosity and permeability estimation, respectively. The accuracy is satisfactory. Comparison analysis of the results from all of the testing datasets demonstrates good performance using the network predictor for the two geophysical parameters, especially for permeability, which is more sensitive to the rocks heterogeneity [30].

Furthermore, in order to testify the advantages of OP-ELM, BP network and support vector machine (SVM) algorithm are used in the model training and testing process for comparison with OP-ELM. Backpropagation feedforward network (BP) is the most commonly used ANN approach, and it is also criticized for having difficulty to decide learning rates, being easy to be stuck on local minimums, having overfit problems, and being time consuming [6]. SVM is a competitive technique which has been intensively used for nonlinear modeling. It has two advantages over traditional deterministic methods: strong nonlinear approximation capabilities and good generalization effectiveness. Experiments have been performed to porosity and permeability prediction in [15–17], which shows better prediction performance than multilayer perceptron (MLP). References [20, 31] have compared performances of BP, ELM, and SVM models, and the experimental results show that ELMs outperform SVMs on reliabilities, while SVMs are better on output distributions. So, it is interesting to compare them in this research.

In the BP network model, the typical structure is used, with three layers including one input layer, one hidden layer, and one output layer. At input layer, 5 neurons stand for the five input logs, and 2 neurons for the output layer. Sigmoid

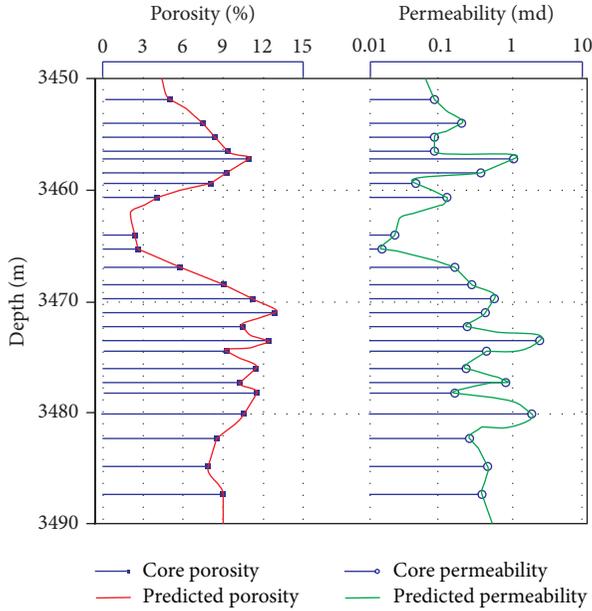


FIGURE 8: Prediction porosity and permeability from OP-ELM network and core values with depth.

kernel function is also adopted for the network model. 33 neurons at the hidden layer are finally determined after dozens of testing in the training process.

As for the SVM predictors, when using the same dataset for model training and testing, minor difference could be created for the SVM models with different kernel function, especially for such numerical approximation problems [31]. But accuracy always needs to be guaranteed, so three types of kernel function have been tested and compared using the dataset of the survey. The three common kernel functions include (1) polynomial function, (2) radial basis function (RBF), and (3) sigmoid function. Here, grid and pattern search methods are used to determine the optimal set of SVM input parameters. Table 1 shows the final results. Gaussian RBF function seems better than the others in performing the estimation, so the final SVM model uses RBF as the kernel function.

Table 2 shows the comparison result using the three types of network predictors with the same testing dataset. Accuracy, MAE, and training time are three factors in comparisons, and the values are obtained by averaging estimations of the samples in well Yq8. The table shows the accuracy, mean absolute error (MAE), and total time in seconds for the three processing approaches, respectively. Good performances have been done using the three optimized network estimators, and best results are achieved by OP-ELM with an accuracy of 95.6%, mean absolute error of 0.205, and fast learning speed of 23 seconds.

Figure 10 shows the prediction porosity and permeability in Permian sandstone of Yq8 by using the three algorithms: BP, SVM, and OP-ELM. All three approaches have conducted good prediction performance for the sandstone reservoir parameters estimation, and the model predictions are very

TABLE 1: Comparison of capabilities of SVM with different kernel functions.

| Kernel function | Kernel parameters                         | Training accuracy (%) | Validation accuracy (%) |
|-----------------|-------------------------------------------|-----------------------|-------------------------|
| Gaussian RBF    | $c = 0.125$ ,<br>$r = 0.125$              | 100                   | 97.93                   |
| Sigmoid         | $c = 1$ ,<br>$r = 0.0625$                 | 100                   | 97.86                   |
| Polynomial      | $c = 0.125$ ,<br>$r = 0.125$ ,<br>$d = 1$ | 100                   | 96.85                   |

TABLE 2: Comparison of porosity prediction performance results on OP-ELM against BP and SVM methodology for well Yq8. The comparison strata belong to the Permian reservoir.

| Algorithm | Accuracy | MAE   | Training time (s) |
|-----------|----------|-------|-------------------|
| BP        | 0.901    | 0.827 | 72                |
| SVM       | 0.935    | 0.639 | 51                |
| OP-ELM    | 0.956    | 0.205 | 23                |

close to the core-measured parameters, which shows the advantages in generalization and prediction of the neural network method. But for the three network predictors, OP-ELM has better accuracy than the other two methods. In the interval of 3375–3390 m, the lithology is dominated by porous and permeable sandstone, with average porosity of about 10% and permeability of more than 1 md.

## 5. Reservoir Parameters Prediction

The above analysis has shown the reliability and accuracy of the OP-ELM prediction model. Therefore the model is then used to estimate porosity and permeability for the remaining wells in the survey. Log data of 7 wells without core data have been input into the model, and porosity and permeability of the Permian reservoir have been estimated. Figure 11 is the plot of original logs and predicted reservoir parameters of well Yq4 from 3560 to 3600 m. This interval belongs to the Permian formation, and oil show exists at the depth of 3591–3598 m. According to the predicted result, porosity of the oil-bearing interval is about 10.78% and permeability is about 3 md, which means the sandstone reservoir from 3590 m to 3598 m has good petrophysical properties and can be recommended for further evaluation.

Since all wells have been processed using the ELM-based model, statistical reservoir analysis is then to be carried out in the survey. According to the geological correlation analysis, tops of target sandstone reservoir in the wells are determined. Thereafter under the constraints of tops boundaries of the reservoir, average porosity and permeability of the target reservoir interval have been calculated for all of 15 wells in the survey. Then flex gridding algorithm is used to interpolate the attributes between wells. Figure 12 is the average porosity map for target reservoir. In the map, red-yellow color stands for high porosity with values bigger than 9.5%. It is obvious

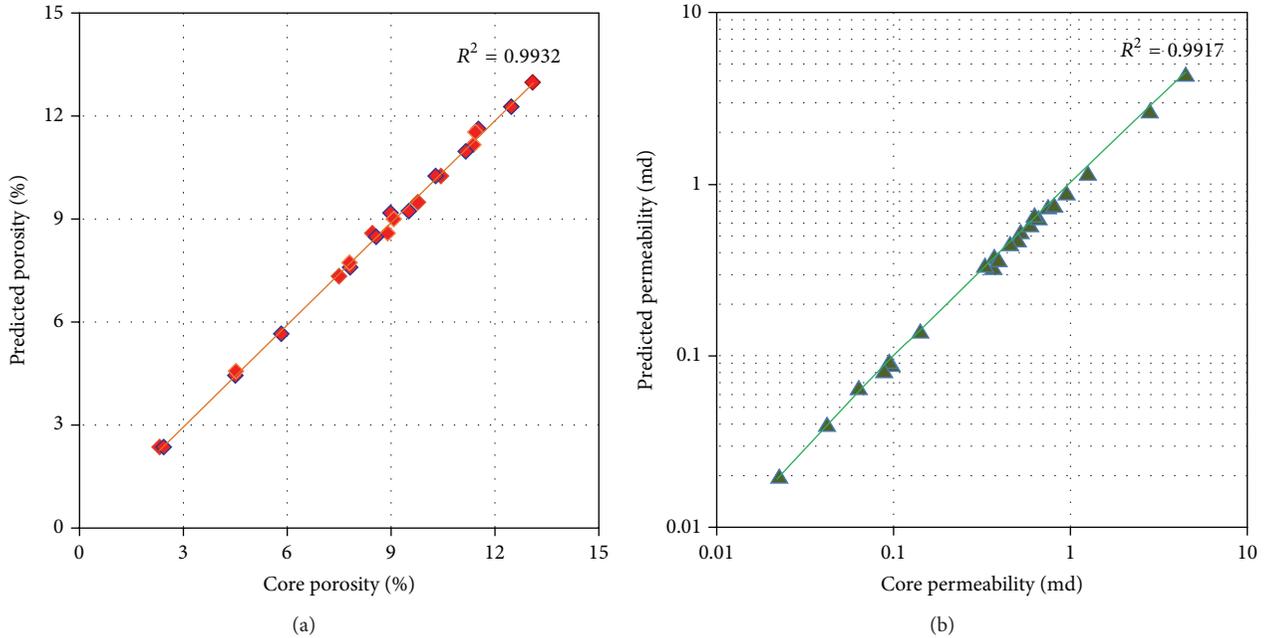


FIGURE 9: Regression analysis of the predicted and core-measured parameters: (a) porosity plot, with correlation coefficient about 0.9932, and (b) permeability plot, with correlation coefficient about 0.9917.

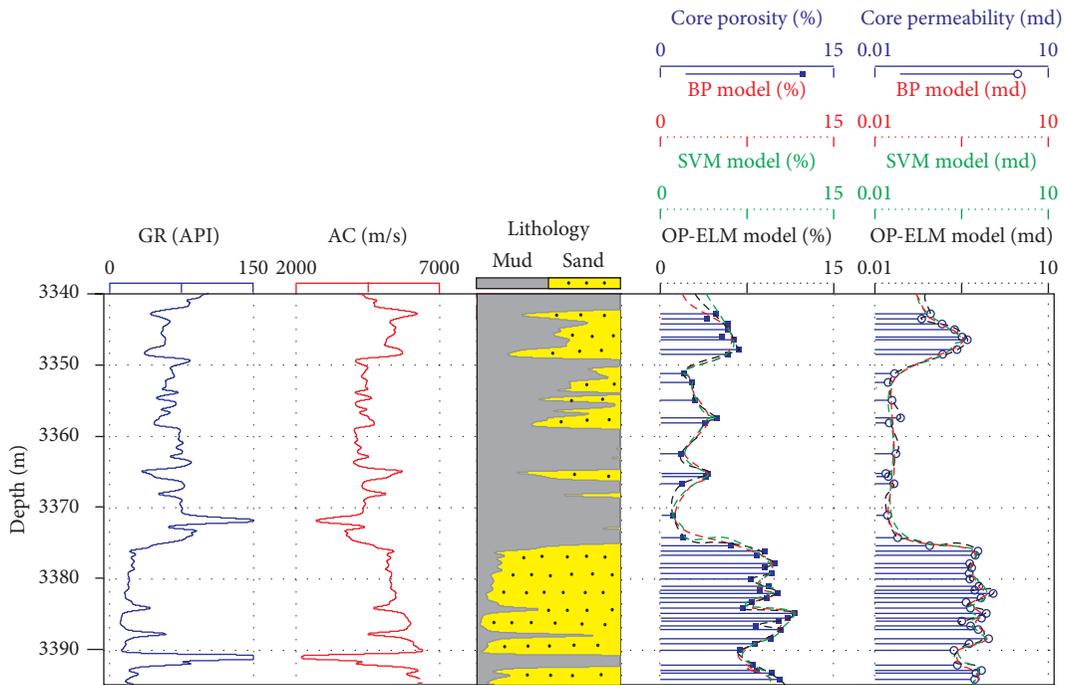


FIGURE 10: Porosity and permeability prediction comparison by three neural network models, BP, SVM, and OP-ELM, for Permian sandstone in Yq8.

that approximate semicircular zones of wells Yq12 and Yq13 at the northeast part of the survey have porosity higher than 10% and also for the triangle zone including wells Yq15, Yq8, Yq4, Yq3, and Yq7 at the western part in the survey. These two areas have better porous and permeable sandstone reservoir, and the six wells included in the two zones have

encountered oil in Permian sandstone reservoir. Since there is no direct structural trap in the survey, stratigraphic trap is the dominant type of traps. Fluvial reservoir is the critical factor for well planning and economic assessment. Therefore the above-mentioned two favorable areas are proposed as potential for next-round well selections in the Yanqi survey.

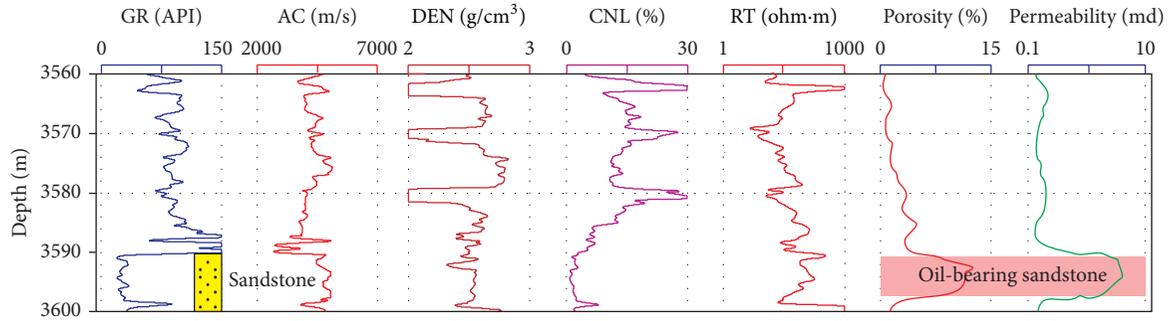


FIGURE 11: Reservoir parameters prediction for well Yq4 in Yanqi survey. Curves of porosity and permeability with blue-dashed line are predicted from the network model. Sandstone reservoir from 3591 m to 3598 m is oil-bearing and has median porosity but high permeability on the basis of the prediction results.

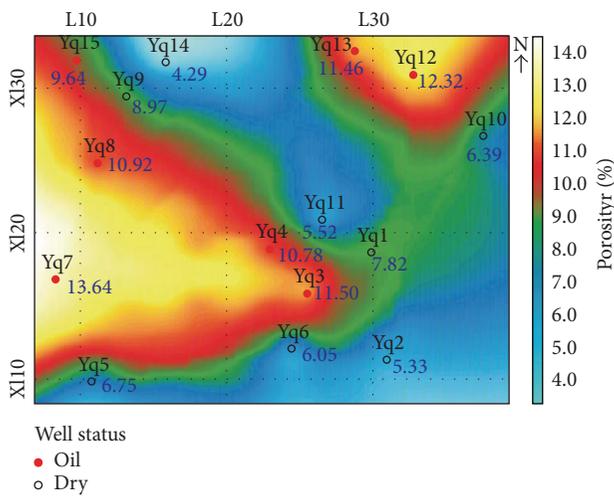


FIGURE 12: Average porosity map for the target reservoir of Yanqi survey. Values near well symbols are the average porosity of Permian sandstone based on the ELM network output.

### 6. Conclusions

The prediction of porosity and permeability is an essential but complex research problem in reservoir characterization of petroleum exploration. In this paper, the authors adopt the ELM-based predictors for solving such crucial problem in heterogeneous sandstone reservoir of Permian formation in Yanqi survey. The proposed ELM and OP-ELM approaches have been reviewed and applied to build estimation models to predict reservoir parameters. Logs data and core-measured porosity and permeability are input into the network model. Sigmoid kernel is used in the network, and the node number of hidden layer is determined by OP-ELM. The advantages of estimation accuracy and learning speed have been testified for ELM methodology in the research. Reliable network prediction models have been established in the study, and porosity and permeability are estimated for Permian sandstone reservoir of all wells in the survey. Potential areas are suggested finally, including two favorable zones: the northeast approximate semicircular zone of wells Yq12 and Yq13 and

western triangle zone including wells Yq15, Yq8, Yq4, Yq3, and Yq7.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

### Acknowledgments

This work is supported by National Natural Science Foundation of China (no. 61402331). The Foundation of Educational Commission of Tianjin City, China (Grant no. 20140803), also funded this research. The authors are also grateful to the three reviewers of the original paper. Jianhua Cao is the first author who wrote the whole paper including text, tables, and figures. Questions can also be sent to the author at the following email: caojh@tust.edu.cn.

### References

- [1] A. K. El Ouahed, D. Tiab, A. Mazouzi, and A. J. Safraz, "Application of artificial intelligence to characterize naturally fractured reservoirs," Paper SPE 84870, 2013.
- [2] G. Wang and T. R. Carr, "Methodology of organic-rich shale lithofacies identification and prediction: a case study from Marcellus Shale in the Appalachian basin," *Computers & Geosciences*, vol. 49, pp. 151–163, 2012.
- [3] L. Qi and T. R. Carr, "Neural network prediction of carbonate lithofacies from well logs, Big Bow and Sand Arroyo Creek fields, Southwest Kansas," *Computers & Geosciences*, vol. 32, no. 7, pp. 947–964, 2006.
- [4] S. Chikhi and M. Batouche, "Probabilistic neural method combined with radial-bias functions applied to reservoir characterization in the Algerian Triassic province," *Journal of Geophysics and Engineering*, vol. 1, no. 2, pp. 134–142, 2004.
- [5] A. K. El Ouahed, D. Tiab, and A. Mazouzi, "Application of artificial intelligence to characterize naturally fractured zones in Hassi Messaoud Oil Field, Algeria," *Journal of Petroleum Science and Engineering*, vol. 49, no. 3–4, pp. 122–141, 2005.
- [6] R. B. C. Gharbi and G. A. Mansoori, "An introduction to artificial intelligence applications in petroleum exploration and

- production,” *Journal of Petroleum Science and Engineering*, vol. 49, no. 3-4, pp. 93–96, 2005.
- [7] A. K. Verma, B. A. Cheadle, A. Routray, W. K. Mohanty, and L. Mansinha, “Porosity and Permeability estimation using neural network approach from well log data,” CSPG/CSEG/CWLS GeoConvention 2012, Search and Discovery Article #41276, 2012.
- [8] M. Ali and A. Chawathé, “Using artificial intelligence to predict permeability from petrographic data,” *Computers & Geosciences*, vol. 26, no. 8, pp. 915–925, 2000.
- [9] M. Baneshi, M. Behzadijo, M. Schaffie, and H. Nezamabadi-Pour, “Predicting log data by using artificial neural networks to approximate petrophysical parameters of formation,” *Petroleum Science and Technology*, vol. 31, no. 12, pp. 1238–1248, 2013.
- [10] H. Khoshdel and M. A. Riahi, “Multi attribute transform and neural network in porosity estimation of an offshore oil field—a case study,” *Journal of Petroleum Science and Engineering*, vol. 78, no. 3-4, pp. 740–747, 2011.
- [11] M. A. Ahmadi, M. Reza, S. M. Hosseini, and M. Ebadi, “Connectionist model predicts the porosity and permeability of petroleum reservoirs by means of petro-physical logs: application of artificial intelligence,” *Journal of Petroleum Science and Engineering*, vol. 123, pp. 643–656, 2014.
- [12] M. Saemi, M. Ahmadi, and A. Y. Varjani, “Design of neural networks using genetic algorithm for the permeability estimation of the reservoir,” *Journal of Petroleum Science and Engineering*, vol. 59, no. 1-2, pp. 97–105, 2007.
- [13] G. L. Jing, W. Du, and Y. Y. Guo, “Studies on prediction of separation percent in electrodialysis process via BP neural networks and improved BP algorithms,” *Desalination*, vol. 291, pp. 78–93, 2012.
- [14] C. Nello and S. T. John, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Publishing House of Electronics Industry, 2004.
- [15] A. F. Al-Anazi and I. D. Gates, “Support vector regression to predict porosity and permeability: effect of sample size,” *Computers & Geosciences*, vol. 39, pp. 64–76, 2012.
- [16] A. F. Al-Anazi and I. D. Gates, “Support vector regression for porosity prediction in a heterogeneous reservoir: a comparative study,” *Computers & Geosciences*, vol. 36, no. 12, pp. 1494–1503, 2010.
- [17] C. Cranganu and M. Breaban, “Using support vector regression to estimate sonic log distributions: a case study from the Anadarko Basin, Oklahoma,” *Journal of Petroleum Science and Engineering*, vol. 103, pp. 1–13, 2013.
- [18] R. Gholami, A. Moradzadeh, M. Shahoo, A. Saman, and J. Hanachi, “Applications of artificial intelligence methods in prediction of permeability in hydrocarbon reservoirs,” *Journal of Petroleum Science and Engineering*, vol. 122, pp. 643–656, 2014.
- [19] U. Iturrarán-Viveros and J. O. Parra, “Artificial Neural Networks applied to estimate permeability, porosity and intrinsic attenuation using seismic attributes and well-log data,” *Journal of Applied Geophysics*, vol. 107, pp. 45–54, 2014.
- [20] H. Zhong, C. Miao, Z. Shen, and Y. Feng, “Comparing the learning effectiveness of BP, ELM, I-ELM, and SVM for corporate credit ratings,” *Neurocomputing*, vol. 128, pp. 285–295, 2014.
- [21] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme learning machine: a new learning scheme of feedforward neural networks,” in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '04)*, vol. 2, pp. 985–990, Budapest, Hungary, July 2004.
- [22] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [23] G.-B. Huang, D. H. Wang, and Y. Lan, “Extreme learning machines: a survey,” *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [24] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [25] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. M. J. Wu, “Human action recognition using extreme learning machine based on visual vocabularies,” *Neurocomputing*, vol. 73, no. 10–12, pp. 1906–1917, 2010.
- [26] R. Moreno, F. Corona, A. Lendasse, M. Graña, and L. S. Galvão, “Extreme learning machines for soybean classification in remote sensing hyperspectral images,” *Neurocomputing*, vol. 128, pp. 207–216, 2014.
- [27] A. Mozaffari and N. L. Azad, “Optimally pruned extreme learning machine with ensemble of regularization techniques and negative correlation penalty applied to automotive engine cold-start hydrocarbon emission identification,” *Neurocomputing*, vol. 131, pp. 143–156, 2014.
- [28] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, “OP-ELM: optimally pruned extreme learning machine,” *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.
- [29] T. Similä and J. Tikka, “Multiresponse sparse regression with application to multidimensional scaling,” in *Artificial Neural Networks: Formal Models and Their Applications—ICANN 2005*, vol. 3697, pp. 97–102, 2005.
- [30] P. Bagheripour, “Committee neural network model for rock permeability prediction,” *Journal of Applied Geophysics*, vol. 104, pp. 142–148, 2014.
- [31] J. Chorowski, J. Wang, and J. M. Zurada, “Review and performance comparison of SVM- and ELM-based classifiers,” *Neurocomputing*, vol. 128, pp. 507–516, 2014.

## Research Article

# Keyword Search over Probabilistic XML Documents Based on Node Classification

Yue Zhao, Ye Yuan, and Guoren Wang

College of Information Science and Engineering, Northeastern University, Liaoning, Shenyang 110819, China

Correspondence should be addressed to Yue Zhao; zhaoy0927@163.com

Received 22 August 2014; Revised 31 October 2014; Accepted 31 October 2014

Academic Editor: Amaury Lendasse

Copyright © 2015 Yue Zhao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper describes a keyword search measure on probabilistic XML data based on ELM (extreme learning machine). We use this method to carry out keyword search on probabilistic XML data. A probabilistic XML document differs from a traditional XML document to realize keyword search in the consideration of possible world semantics. A probabilistic XML document can be seen as a set of nodes consisting of ordinary nodes and distributional nodes. ELM has good performance in text classification applications. As the typical semistructured data; the label of XML data possesses the function of definition itself. Label and context of the node can be seen as the text data of this node. ELM offers significant advantages such as fast learning speed, ease of implementation, and effective node classification. Set intersection can compute SLCA quickly in the node sets which is classified by using ELM. In this paper, we adopt ELM to classify nodes and compute probability. We propose two algorithms that are based on ELM and probability threshold to improve the overall performance. The experimental results verify the benefits of our methods according to various evaluation metrics.

## 1. Introduction

Traditional databases only manage deterministic information, but many applications use databases to involve uncertain data such as information extraction, information integration, and web data mining. Because of the flexibility of XML data model, it can easily allow a natural representation of uncertain data. Now, many probabilistic XML models are designed and analyzed [1–4]. This paper selects a popular probabilistic XML model  $\text{PrXML}_{\{\text{ind}, \text{mux}\}}$  [5], which is discussed in [6]. In this model, a probabilistic XML document (called a  $p$ -document) is considered as a labeled tree which has two types of nodes, *ordinary* nodes and *distributional* nodes. Ordinary node is used to represent the actual data and distributional node is used to represent the probability distribution of the child nodes. There are two types of distributional nodes, IND and MUX. If a node is an IND node, its children nodes are *independent* of each other, while the children of a MUX node are *mutually exclusive*; that means, at most, one child can exist in a random instance document (a *possible world*). A real number from  $(0, 1]$  is attached on each edge in an XML tree, indicating the conditional probability that the child node will appear under the parent node given the existence of its

father node. From the attribute of a MUX node, we can see that the sum of all the existence probabilities of children nodes is 1 or less than 1.

Keyword search has been widely applied on XML data. It is considered to be an effective information discovery method to query XML data. Users do not need know the knowledge of the underlying data structures and complex query language beforehand. So, keyword search is an easy method for ordinary users to retrieve information. Keyword search on XML data is different from the query on text data. As a result, a subtree rooted at a common ancestor node will replace the whole text data. In the past years, the definition of a common ancestor node has several choices, such as LCA (lowest common ancestor), SLCA (smallest LCA), and ELCA (exclusive LCA). These definitions are used to determine the users' query intentions. SLCA and ELCA are the subset of LCA by adding some restrictive factor. In many cases, the size of a set determines the accuracy of the query. This paper selects SLCA as the root node of result subtree because that SLCA nodes set is the smallest set in all the definitions based on LCA.

It is known that both neural networks and SVM (*support vector machines*) have been playing the dominant roles out of numerous computational intelligence techniques.

But they face three challenging issues such as (1) slow learning speed, (2) trivial human intervene, and (3) poor computational scalability. ELM [7, 8] as emergent technology works for generalized single-hidden layer feedforward networks (SLFNs). ELM [9–12] has good performance on classification applications and can be used to classify nodes before query XML data. Classification is considered as an important cognitive computation task [13–16]. An XML data tree can be seen as a set of all the nodes including root node (only one), connected nodes, and leaves nodes. A connected node has only one father node and one or more children nodes. The keyword usually appears in the leaves nodes or its father node of a leaf node. So, the classification needs to consider two kinds of information, and they are keyword information and structural information. XML contains some structural information such as the element-subelement relationships and the element-value relationships. The element-subelement includes ancestor-descendant relationship, and father-child relationship. In addition, sibling relationship is an important relationship. If the number of keywords is more than one, the relationship between nodes plays a crucial role in the keyword search on XML data. So, the classification needs to think out structural information and keyword information on an XML data tree. The classification method is presented in Section 3.

This paper is organized as follows. Section 2 introduces the probabilistic XML model and the formal semantics of keyword search result on probabilistic XML data. Section 3 shows how to classify nodes and the calculation method of probability. In Section 4, we propose an algorithm to query keyword on probabilistic XML data by using ELM to classify nodes. Section 5 introduces the method which describe the impact of the probability threshold. The experimental and performance evaluations are presented in Section 6. Sections 7 and 8 give the related work and the conclusion.

## 2. Problem Definitions

**2.1. Probabilistic XML Data.** A probabilistic XML document ( $p$ -document) can be seen as a set of many deterministic XML documents. Each deterministic document is called a possible world. A probabilistic XML document represented as a labeled tree has *ordinary* nodes representing actual data and *distributional* nodes representing the probability distribution of the child nodes. Ordinary nodes are prime XML nodes and they always appeared on deterministic XML data and probabilistic XML data. Distributional nodes are only used to define the probabilistic process of generating deterministic documents, while those nodes do not occur on deterministic XML data. This paper adopts  $\text{PrXML}^{\{\text{ind}, \text{mux}\}}$  as the probabilistic XML model. For example, Figure 1 shows a  $p$ -document  $T$ . Ordinary nodes are shown as a black solid point, for example, {lab}, {manager}, {Tom} and {Tony}. IND nodes are depicted as rectangular boxes with rounded corners, for example, IND1, IND2, and IND3. MUX nodes are displayed as circles, for example, MUX1.

A  $p$ -document can generate all possible worlds (deterministic documents). Given a  $p$ -document  $T$ , we can traverse

$T$  in a top-down fashion. When we visit a distributional node, there are two situations according to the different types. One situation is that if a node is an IND node with  $m$  children nodes, we generate  $2^m$  copies. We randomly select children nodes of the IND node into a copy. A copy is a subset of all children nodes. For each copy, the probability is the product of all existence probabilities of the children nodes in the subset and the absence probabilities ( $1 - \text{existence probability}$ ) of the children nodes not in the subset. Another situation is that if a node is a MUX node with  $m$  children nodes, we generate  $m$  or  $m + 1$  copies. If the sum of all the existence probabilities of the children nodes is 1, there are  $m$  copies, otherwise the number of copies is  $m + 1$ . For each copy, the probability is the existence probability of the selected children node. If none of the children nodes has been selected, the probability is the absence probability. For example, Figure 2 shows the copies of a  $p$ -document with their probabilities. Figure 2(a) select node  $b$  as the only child node of node  $a$ , and the probability is  $0.7 * (1 - 0.6) = 0.28$ . If there is not any node selected as the children nodes of  $a$ , Figure 2(d) shows the probability of this copy is  $(1 - 0.7) * (1 - 0.6) = 0.12$ . As shown in Figure 2(e), node  $a$  selects nodes  $b$  and  $c$  as its children nodes, and node  $c$  selects node  $d$  as its child node. The probability is  $0.7 * 0.6 * 0.5 = 0.21$ . The probabilities of the other copies (*possible worlds*) are easy to calculate from the above procedure.

**2.2. Keyword Query.** Usually, we model an XML tree as a labeled ordered tree, in which nodes represent elements and edges represent direct nesting relationship between nodes. Recently, keyword search has been studied in XML documents more and more. Given a set of keywords and an XML document, most work took LCA and SLCA of the matched nodes as the results. The function  $\text{lca}(v_1, v_2, \dots, v_k)$  computes the Lowest Common Ancestor of nodes  $v_1, v_2, \dots, v_k$ . Given  $k$  keywords and the inverted lists  $\{S_1, S_2, \dots, S_k\}$  of them. The LCA of these keywords on  $T$  is defined as

$$\begin{aligned} \text{lca}(v_1, v_2, \dots, v_k) \\ &= \text{lca}(S_1, S_2, \dots, S_k) \\ &= \{\text{lca}(n_1, n_2, \dots, n_k) \mid n_1 \in S_1, \dots, n_k \in S_k\} \end{aligned} \quad (1)$$

$\text{child}(v, n_i)$  denote the child nodes of node  $v$  on the path from  $v$  to  $n_i$ .

**Definition 1** (SLCA on XML data). Given a query  $Q$  in an XML tree  $T$ , an SLCA query finds the SLCA nodes  $v$  which is the child node of other LCA nodes.

The SLCA is defined as follows:

$$\begin{aligned} \text{slca}(\{v_1\}, S_2, \dots, S_k) \\ &= \{v \mid v \in \text{lca}(\{v_1\}, S_2, \dots, S_k), \\ &\quad \forall v' \in \text{lca}(\{v_1\}, S_2, \dots, S_k) (v \neq_a v')\}. \end{aligned} \quad (2)$$

For example, Figure 3(a) gives a traditional XML tree which is generated by Figure 1 and a query  $Q = \{\text{Tom}, \text{XML}\}$ .

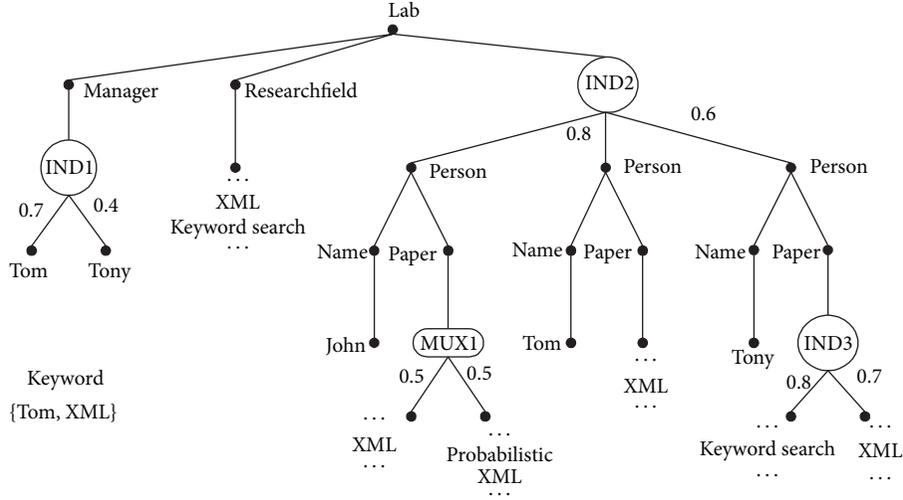


FIGURE 1: A probabilistic XML document.

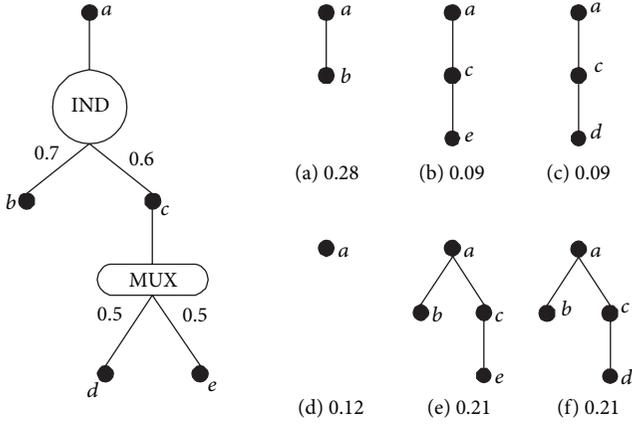


FIGURE 2: A probabilistic subtree.

The result of this query is shown in Figure 3(b). The node lab and person are LCA nodes. The SLCA node is person. According to the concept of SLCA, the node lab is an ancestor node of person. So, the SLCA node is person. From the definition of LCA and SLCA, we can see that SLCA is a subset of LCA.

This paper selects SLCA as the result for the keyword search on probabilistic XML data. Because SLCA is the smallest set, every SLCA node should be seen as a suitable result for users. SLCA node is the smallest common node of the nodes which contain keywords. As the result, we return the subtree rooted at SLCA node. So, when classification algorithm classify the nodes, it need put the keyword node and its all child nodes into one set.

A keyword search on  $p$ -document consists of a  $p$ -document  $T$ , a query  $Q = \{k_1, k_2, \dots, k_n\}$ . We define the answer for a keyword search on  $T$  as ordinary nodes on  $T$  to be SLCA in the possible worlds which is generated by  $T$ . The probability of a node  $v$  being an SLCA in the possible

worlds is denoted as  $\Pr_{\text{slca}}^T(v)$ . The formal definition is shown as follows:

$$\Pr_{\text{slca}}^T(v) = \sum_{i=1}^m \{\Pr(w_i) \mid \text{slca}(v, w_i) = \text{true}\}, \quad (3)$$

where  $\Pr(w_i)$  is the existence probability of the possible world  $w_i$ .  $\{w_1, w_2, \dots, w_n\}$  denotes the possible worlds generated by  $T$ .  $\text{slca}(v, w_i) = \text{true}$  indicates that  $v$  is an SLCA in the possible world  $w_i$ .

$\Pr_{\text{slca}}^T(v)$  can also be computed with (4). Here,  $\Pr(\text{path}_{r \rightarrow v})$  indicates the existence probability of  $v$  in the possible worlds. We can compute the existence probability by multiplying the conditional probabilities along the path from the root node  $r$  to node  $v$ .  $\Pr_{\text{slca}}^{\text{sub}}(v)$  is the local probability for  $v$  being an SLCA node in  $T_{\text{sub}}(v)$ .  $T_{\text{sub}}(v)$  denotes a subtree of  $T$  rooted at  $v$ . Consider

$$\Pr_{\text{slca}}^T(v) = \Pr(\text{path}_{r \rightarrow v}) \times \Pr_{\text{slca}}^{\text{sub}}(v). \quad (4)$$

From (3), we can obtain the following equation to compute  $\Pr_{\text{slca}}^{\text{sub}}(v)$ :

$$\Pr_{\text{slca}}^{\text{sub}}(v) = \sum_{i=1}^{m'} \{\Pr(w'_i) \mid \text{slca}(v, w'_i) = \text{true}\}, \quad (5)$$

where  $\{w'_1, w'_2, \dots, w'_m\}$  denotes the local possible worlds generated from  $T_{\text{sub}}(v)$ .  $\text{slca}(v, w'_i) = \text{true}$  means that  $v$  is an SLCA in the possible world  $w'_i$  rooted at  $v$ ; namely, the root node  $v$  is the only LCA node. The probability of this subtree rooted at node  $v$  can be shown as

$$\Pr_{\text{slca}}^{\text{sub}}(v) = \Pr(\text{path}_{v \rightarrow k_1}) \times \Pr(\text{path}_{v \rightarrow k_2}). \quad (6)$$

As an SLCA result of keyword search, the probability of node  $v$  is

$$\begin{aligned} \Pr_{\text{slca}}^T(v) &= \Pr(\text{path}_{r \rightarrow v}) \times \Pr_{\text{slca}}^{\text{sub}}(v) \\ &= \Pr(\text{path}_{r \rightarrow v}) \times \Pr(\text{path}_{v \rightarrow k_1}) \times \Pr(\text{path}_{v \rightarrow k_2}). \end{aligned} \quad (7)$$

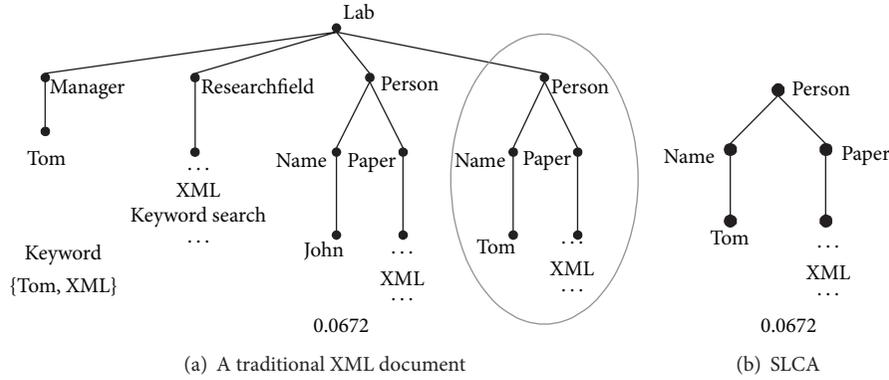


FIGURE 3: SLCA nodes on XML data.

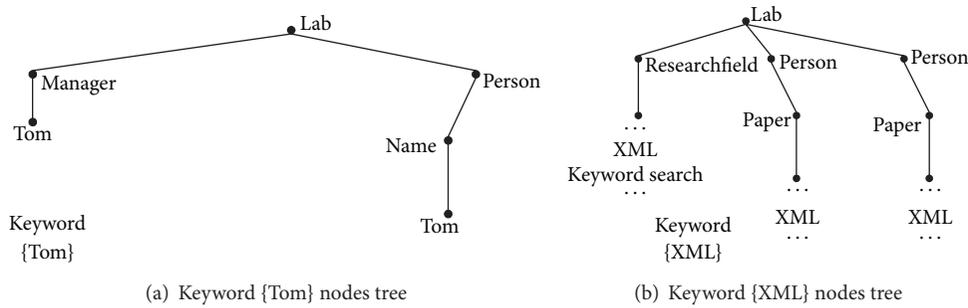


FIGURE 4: Keyword nodes tree.

For example, we give an example to compute  $\Pr_{\text{slca}}^T(\text{person})$  (we select the node person with the existence probability of 0.8) in Figure 1. We can see that  $\Pr(\text{path}_{\text{lab} \rightarrow \text{person}}) = 0.8$ .  $\Pr_{\text{slca}}^{\text{sub}}(\text{person}) = 1$ . So, the result is  $\Pr_{\text{slca}}^T = 0.8 \times 1 = 0.8$ .

**Definition 2** (SLCA on probabilistic XML data). Given a query  $Q$  in a probabilistic XML tree  $T$ , an SLCA query finds the SLCA nodes  $v$  in all possible worlds with the probability of all the probabilities of the possible worlds in which the node  $v$  is an SLCA node.

**Definition 3** (Threshold SLCA on probabilistic XML data). Given a query  $Q$  in a probabilistic XML tree  $T$ , an SLCA query finds the SLCA nodes  $v$  in all possible worlds with the probability of all the probabilities of the possible worlds in which the node  $v$  is an SLCA node. And the probability must be more than threshold  $\theta$ .

If we do not consider the distributional nodes of the  $p$ -document, we can see that every SLCA node is an LCA node. For a given keyword query  $Q$  and a XML tree, node  $v$  is a common ancestor of  $Q$  on an XML tree if the subtree rooted at  $v$  contains each keyword of  $Q$  at least once. For example, for query  $Q = \{\text{Tom}, \text{XML}\}$ , the common ancestor nodes of  $Q$  are lab and person. If we built a tree contains common ancestor nodes according to the relationship on a XML tree, the leaves nodes are the SLCA nodes.

### 3. Classification of Nodes and Calculation of Probability

A  $p$ -document contains two types of nodes, ordinary nodes and distributional nodes. An ordinary node can represent actual data on probabilistic XML data, but a distributional node can only represent the probability distribution of a node.

**3.1. Classification of Ordinary Nodes.** From Section 2, we can see that if we can find keyword nodes tree, the set intersection operation for keyword nodes tree should achieve SLCA nodes quickly. Figures 4(a) and 4(b) show the keyword nodes tree. When we use set intersection operation to obtain the common ancestor nodes tree such as shown in Figure 5. So, the important section is how to receive the keyword nodes tree.

To receive the keyword nodes tree, we need to add dummy nodes for actual nodes which contain more than one keyword. If the subtree rooted at the node  $v$  contains two keywords, we should add one dummy node as the sibling node of node  $v$ . For example, node {lab} and {person} in Figure 5 has its dummy node. These dummy nodes do not exist in the actual tree. The aim of adding dummy nodes is to classify nodes effectively.

**3.2. Classification of Distributional Nodes.** Distributional nodes can represent the probability distribution of the children nodes. A  $p$ -document defines a probability distribution

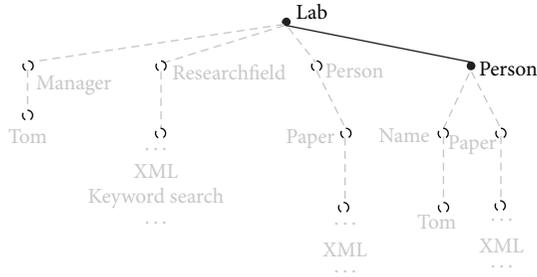


FIGURE 5: A common nodes tree.

over a space of deterministic XML documents. According to the different types of the distributional node, the number of copies is different.

*Case 1* (If a node is an IND node). It has  $n$  children nodes, the number of copies is  $2^n$ . If there is only one child node  $v_1$  contains keyword  $k_1$  and the probability of  $v_1$  is  $p_{v_1}$ , the probability of the subtree rooted as  $v_1$  which contains  $k_1$  is  $p_{v_1}$ . If the number of children nodes which contains keyword  $k_1$  is  $n$ , the probability need consider all the situations which contains keyword  $k_1$ . The sum of these probabilities is the probability of the subtree which contains keyword  $k_1$ . Considering all the situations is a very complicated calculation process. So, we can first calculate the probability of the subtree which do not contain the keyword  $k_1$ . So, the probability of an IND node is shown as (8).

$$p_{k_1}(v) = \begin{cases} p_{v_1} & \{\text{one } k_1 \text{ matched node}\} \\ 1 - \prod_{i=1}^n (1 - p_{v_i}) & \{\text{n } k_1 \text{ matched nodes}\} \end{cases} \quad (8)$$

*Case 2* (If a node is a MUX node). The number of child nodes is  $n$  or  $n + 1$ . Each copy has its probability value. Some of the copies will contain the keyword, and the copies which contain the keyword are important for our probabilistic keyword search. So, the probability of a MUX node is shown as (9).

$$p_{k_1}(v) = \sum_{i=1}^n p_{v_i}. \quad (9)$$

Figure 6(a) shows an example of an IND node. For the keyword {Tom}, IND1 is a father node of nodes {Tom} and {Tony}. The copy with the existing of {Tom} has two situations, and their probabilities are  $0.7 \times (1 - 0.4) = 0.42$  and  $0.7 \times 0.4 = 0.28$ . It means that the probability of the subtree rooted at node IND1 contains the keyword {Tom} is  $0.42 + 0.28 = 0.7$ . Figure 6(b) shows an example of a MUX node. For the keyword {XML}, MUX1 is a parent node of node {XML} and {Probabilistic XML}. The copy with the existing of {XML} has two situations; the probabilities are all 0.5. It means that the probability of the subtree rooted at node MUX1 containing the keyword {XML} is  $0.5 + 0.5 = 1$ .

For each keyword, all its ancestor nodes and itself nodes will constitute a tree. This tree contains ordinary nodes and distributional nodes. To present the probability contribution situation of this tree which contains keywords, we will delete

distributional nodes and connect its children nodes to its father node with the existence probability of containing the keyword of the subtree rooted at its father node according to the type of distributional node. For example, Figure 7(a) shows a tree contained keyword Tom. Node {manager} is a father node of node {IND1}. The probability of a subtree rooted at node {IND1} which contains keyword Tom is 0.7. As shown in Figure 7(b), it is the situation of the probabilistic tree which contains keyword XML.

We merge all the keywords probabilistic trees together. It will generate a keyword nodes probabilistic tree. We need calculate SLCA nodes on this tree with the probability and delete the subtree rooted at SLCA nodes. Next, we need to continue to calculate SLCA results on remaining nodes tree. So, if we repeat such operation, all the SLCA results will be generated. For example, Figure 8 is a keyword nodes probabilistic tree. This tree retains all the probabilities of the subtree which contains keyword.

*3.3. Probability Calculation.* If we calculate SLCA results on the tree in Figure 8, the node {person} which is shown in Figure 9(a) is the only result. So, we need to delete the subtree which is rooted at node {person}, and the remaining nodes tree is shown in Figure 9(b). To repeat calculated operation on the remaining nodes tree, we can see that the node {lab} is another result. In this section, we introduce how to calculate the probabilities of all the SLCA nodes.

A distributional node can appear in all the positions in a tree excepted root node and leaf nodes. If node  $v$  is an SLCA node and there are two keywords  $\{k_1, k_2\}$ , the probability of node  $v$  needs to compute 3 values from (7); they are  $p(\text{path}_{r_1 \rightarrow v})$ ,  $p(\text{path}_{v \rightarrow k_1})$  and  $p(\text{path}_{v \rightarrow k_2})$ . As shown in Figure 10,  $r_1$  represents the path from root node to node  $v$  and  $r_2$ ,  $r_3$  express the path from node  $v$  to  $k_1$ ,  $k_2$ , respectively. So, if there are  $n$  keywords, we need to divide this tree into  $n + 1$  parts.

Next, we discuss how to compute probability according to 4 situations.

*Case 3* ( $r_1$ ,  $r_2$  and  $r_3$  do not contain distributional nodes). If node  $v$  is an SLCA node and there are no distributional nodes in  $r_1$ ,  $r_2$  and  $r_3$ , that means this situation is the same as deterministic XML data. The probability is 1.

*Case 4* ( $r_1$  contains distributional nodes). If  $r_1$  contains distributional nodes, that means node  $v$  will not appear in some possible worlds. These possible worlds do not contain node  $v$ . All the distributional nodes in  $r_1$  will influence other nodes' probability in a keyword nodes probabilistic tree (just as in Figure 8). Figure 11 shows the processing of calculation in a keyword nodes probabilistic tree. The path from node {lab} to node {person} contains a distributional node, and the probability of the path is 0.8. So, when we finish the calculation of node {person}, the probability of node {lab} needs to add a new probability  $1 - 0.8$  as shown in Figure 11.

*Case 5* ( $r_2$  or  $r_3$  contains distributional nodes). If the path from node  $v$  to any keyword matched node contains distributional nodes, it illustrates that this keyword matched

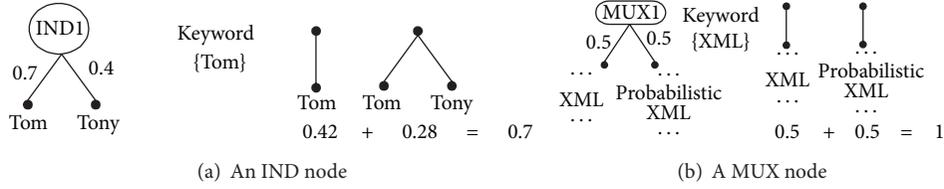


FIGURE 6: Distributional nodes.

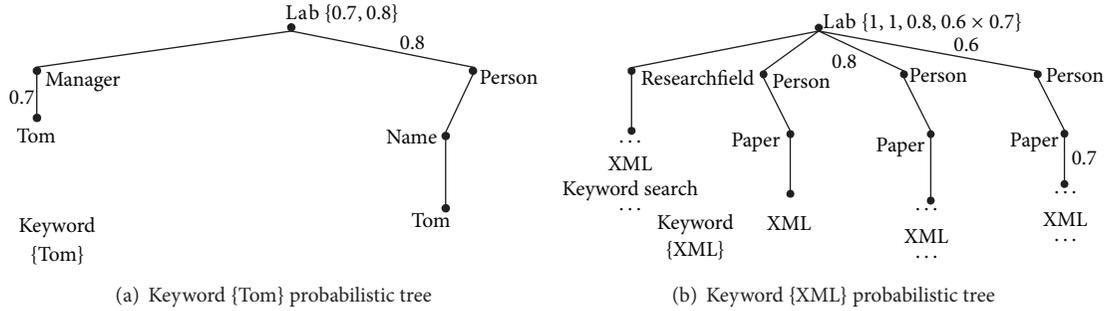


FIGURE 7: Keyword nodes probabilistic tree.

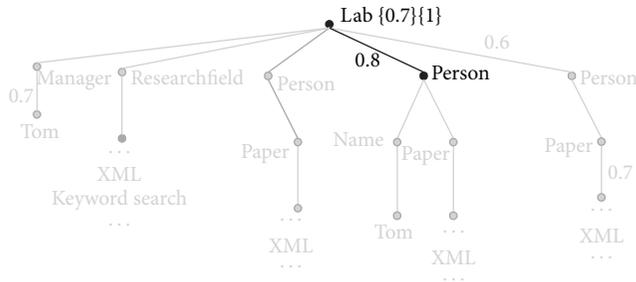


FIGURE 8: A keyword nodes probabilistic tree.

node will not appear in any possible worlds. In this situation, each probability of keyword matched node all need be recorded, just as in Figure 7(a) and Figure 7(b). Figure 7(a) shows the keyword {Tom} probabilistic tree. The node {lab} has two branches; one is the path which constitutes nodes {lab} {manager} and {Tom}, and another one is the path that contains nodes {lab} {person} {name} and {Tom}. The probabilities in these two paths are 0.7 and 0.8 as shown in Figure 7(a). The keyword {XML} is the same as shown in Figure 7(b). Figure 8 is a keyword nodes probabilistic tree. For the first keyword {Tom} in the second path ({lab} {person} and {name}), the node {person} contains another keyword XML. So, the node {lab} only need record the probability 0.7. For the second keyword XML, because the probability of the node {lab} containing it has the situation with value 1; other probabilities do not have to be recoded as shown in Figure 7(b).

Case 6 ( $r_1$   $r_2$  and  $r_3$  all contains distributional nodes). This situation can be seen as the synthesis of two kinds of aforementioned circumstances. The processing method has been introduced in the previous illustration, and in this section we only need to judge which one we need to compute firstly.

Because SLCA nodes are the child nodes in all the common ancestor nodes, SLCA results need to be computed according to the path from bottom to up. In Figure 10, the nodes in  $r_2$  and  $r_3$  are child nodes of the nodes in  $r_1$ . The situation of  $r_2$   $r_3$  has priority right. Next, the situation  $r_1$  needs to be computed as the second step.

#### 4. ELM Keyword Search on Probabilistic XML Data

Keyword search on probabilistic XML data based on classification mainly include four steps, they are shown as follows: (1) adding dummy nodes according to the number of keywords, (2) to classify nodes with ELM based on keyword according to the type of the nodes, (3) using the set merging operation to structure the common ancestor nodes probabilistic tree, and (4) repeat the operation of calculating SLCA and deleting the subtree; all the SLCA results will generate. The key of the keyword search on  $p$ -document is how to calculate the probability of the SLCA results. Step (2) and Step (4) all contain the computation of the probabilities.

Each node contains two kinds of information, they are code and keyword it contained. If a node is a distributional node, there are the third information in this node, that is, probability. Code is used to judge the relationships between nodes, such as finding the common ancestor nodes. The keyword which is contained in a node is the key of keyword search. When we use ELM to classify nodes, keyword can be used as the label of the classification set. Every set represents one keyword. For the given query, we will find all the sets of keywords which is given by users and operate set merging to obtain a keyword nodes tree.

Next, we introduce four steps of the keyword search algorithm using ELM to classify nodes on probabilistic XML data one by one.

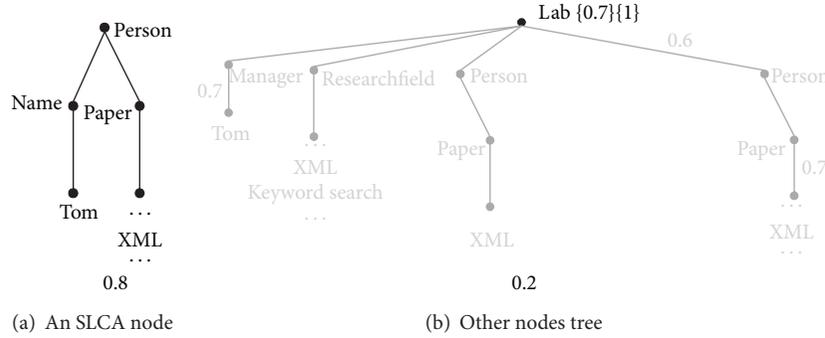


FIGURE 9: SLCA of a common nodes probabilistic tree.

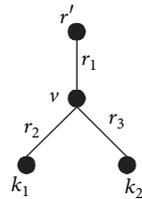


FIGURE 10: Node  $v$  tree.



FIGURE 11: Calculation SLCA probability.

TABLE 1: The label of the nodes.

| Node   | Dummy node | Keyword    |
|--------|------------|------------|
| lab    | {lab}      | {Tom, XML} |
| IND2   | {IND2}     | {Tom, XML} |
| person | {person}   | {Tom, XML} |

First, adding dummy nodes according to the number of keywords. We can see that the probabilistic XML tree in Figure 1 contains two keywords {Tom, XML}. The algorithm uses Dewey code to encode the XML tree. So, the first step is adding the dummy nodes for the node  $v$  which contains keywords in the subtree rooted at the node  $v$ . If the subtree which is rooted at the node  $v$  has  $n$  keywords, we will add  $n - 1$  dummy nodes. For example, in Figure 1, the node {lab}, {IND1}, and node {person} are added into the XML tree as a part of the dummy nodes tree. Each node on the probabilistic XML tree has a table. As shown in Figure 9. From the number of the nodes which contains keyword, the dummy nodes for adding are shown in Table 1.

Second, to classify nodes with ELM based on keyword according to the type of the nodes, from the dummy nodes tree, all the nodes and the distributional nodes consist of the classified nodes. ELM can classify the nodes to two sets such as shown in Figure 12(a). The first set represents keyword Tom, and the second set represents keyword XML. Each

distributional node has a probability which represents the keyword probability of the subtree which is rooted at the distributional node. For example, the node IND1 has the probability with 0.7, that means the probability of containing keyword Tom of the subtree which is rooted at IND1 is 0.7.

Then, we need to delete all the distributional nodes and connect all their children nodes to their parent node. The probability of distributional node will be moved to its child node. For example, in Figure 12(b), the node {Tom} accepts the probability 0.7 from its father node {IND1}.

Third, use the set merging operation to structure the common ancestor nodes probabilistic tree. The intersection of the two sets is the set which includes node {lab} and {person}. Figure 13 shows the union set of two keyword sets.

Finally, repeat the operation of calculating SLCA and deleting the subtree; all the SLCA results will generate. Let us calculate SLCA result on the tree which is shown in Figure 13. The node {person} with the SLCA probability of 0.8 is generated. So, the subtree which is rooted at {person} will be deleted, and the probability  $1 - 0.8 = 0.2$  will be leaves to the other nodes tree. Next, the node {lab} is another result. The probability of this result is  $0.2 \times 0.7 = 0.14$ . Because, the extensive probability of the node Tom is 0.7, and the extensive probability of node {XML} is 1; the SLCA probability of {lab} is 0.14.

From (4) we can see that the product of the extensive probability and the SLCA probability is the result probability. The extensive probability is 1 to the node {person}, so the result probability is  $0.7 \times 1 = 0.7$ . Moreover, the extensive probability of the node {lab} is 1, and the result probability of the node {lab} is  $1 \times 0.14 = 0.14$ .

## 5. ELM-Threshold Keyword Search on Probabilistic XML Data

Pruning can speed up the retrieval speed, we can delete any nodes that are not SLCA nodes in the processing of calculation SLCA results. According to the definition of SLCA, if a node is an SLCA node, its all ancestor nodes are not SLCA nodes. If there are no distributional nodes in  $r_1$ ,  $r_2$  and  $r_3$ , node  $v$  is an SLCA node and its father node  $r'$  is not SLCA node. For example, node {person} is an SLCA node, so its father node {lab} is not an SLCA result.

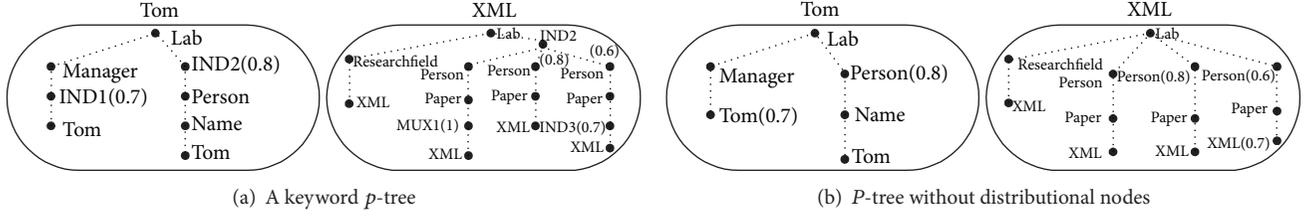
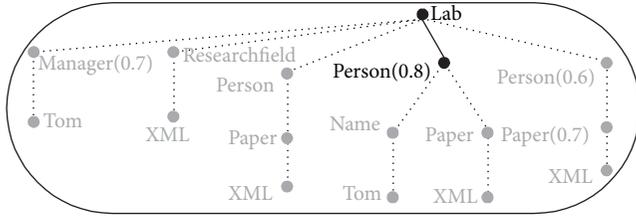


FIGURE 12: Probabilistic tree.

FIGURE 13: A common nodes  $p$ -tree.

Some results' probability is very small, these results have very low extensive probability. For most users' query intension, if the probability is more than other nodes, that means that the node has high query value. So, this section introduces the concept of probability threshold.

*Case 7* ( $r1$  contains distributional nodes). Because node  $v$  is an SLCA node, when we finish the calculation of node  $v$ , its ancestor nodes need to add the probability  $1 - p(v)$ . As shown in Figure 9, the value  $(1 - 0.8) = 0.2$  has been retained in the other nodes tree in Figure 9(b). So, if the probability value  $1 - p(v)$  is less than the probability threshold, the probability of node {lab} must be less than the probability threshold. All the ancestors must be deleted, because they are not threshold SLCA nodes.

*Case 8* ( $r2$  or  $r3$  contains distributional nodes). There are distributional nodes in  $r2$  or  $r3$ . When we finish the calculation of node  $v$  and 2 keywords, there are  $3 = 2^2 - 1$  situations will be remained. We need to consider all these 3 situations. Each situation has the probability  $1 - p_{\text{path}}(v \rightarrow k_1)$ . If the probability value  $1 - p_{\text{path}}(v \rightarrow k_1)$  is less than the probability threshold, the probability of node {lab} must be less than the probability threshold. All the ancestors must be deleted, because they are not threshold SLCA nodes. If there are no  $k_1$  and  $k_2$  in the subtree rooted at  $v$ , the probability is  $(1 - p_{\text{path}}(v \rightarrow k_1)) \times (1 - p_{\text{path}}(v \rightarrow k_2))$ . If the probability value  $(1 - p_{\text{path}}(v \rightarrow k_1)) \times (1 - p_{\text{path}}(v \rightarrow k_2))$  is less than the probability threshold, the probability of node {lab} must be less than the probability threshold. All the ancestors must be deleted, because they are not threshold SLCA nodes.

For a probability threshold  $\theta$ , when we calculate the probability on the keyword nodes probabilistic tree, if the probability is less than the threshold  $\theta$ , the result is not an SLCA node. For example, if the probability threshold is 0.3, the SLCA probability of the second result is  $0.2 \times 0.7 =$

TABLE 2: Properties of probabilistic XML data.

| ID    | Name  | Size  | Ordinary  | IND     | MUX     |
|-------|-------|-------|-----------|---------|---------|
| DOC 1 | XMARK | 10 M  | 159,307   | 14,630  | 15,471  |
| DOC 2 | XMARK | 20 M  | 364,199   | 41,251  | 38,100  |
| DOC 3 | XMARK | 40 M  | 689,470   | 77,228  | 61,535  |
| DOC 4 | XMARK | 80 M  | 1,497,433 | 161,277 | 159,495 |
| DOC 5 | DBLP  | 20 M  | 361,370   | 68,345  | 70,790  |
| DOC 6 | DBLP  | 40 M  | 731,561   | 238,450 | 227,540 |
| DOC 7 | DBLP  | 80 M  | 1,477,345 | 440,007 | 405,857 |
| DOC 8 | DBLP  | 160 M | 3,260,109 | 788,367 | 771,320 |

0.14 which is less than the probability threshold; the node {lab} is not a threshold SLCA node. When we compute the probability of the first result of the node {person}, because the probability  $1 - 0.8 = 0.2$  will be leaves to the other nodes tree, and the probability value is less than the probability threshold, the probability of other nodes tree must be less than the probability threshold. So, we need not calculate SLCA on the other nodes tree. When the first result is computed, the threshold SLCA is finished.

## 6. Performance Verification

In this section, the performance of keyword search used ELM to classify on probabilistic XML data is shown as follows. All the experiments based on ELM for classification algorithms are carried out in MATLAB 2007 environment running in a Pentium 4, 2.53 GHZ CPU.

The dataset we used is shown in Table 2. In this paper, the algorithm selects two datasets XMARK and DBLP. For each XML dataset used, we generate the corresponding probabilistic XML tree, using the same method as used in [5]. Table 3 shows the dataset and keywords. We visit the nodes in the original XML tree in preorder way. For each node  $v$  visited, we randomly generate some distributional nodes as children of  $v$ . For the original children of  $v$ , we select them as the children of the new generated distributional nodes and assign them random probability distributions. We need a restriction that the sum of children nodes for a MUX node is no more than 1. The keyword has 8 situations. The number of keywords is 2 to 3.

Compared with those traditional computational intelligence techniques, ELM provides better generalization performance at a much faster learning speed and with least human intervention. We compare the query times of two situations

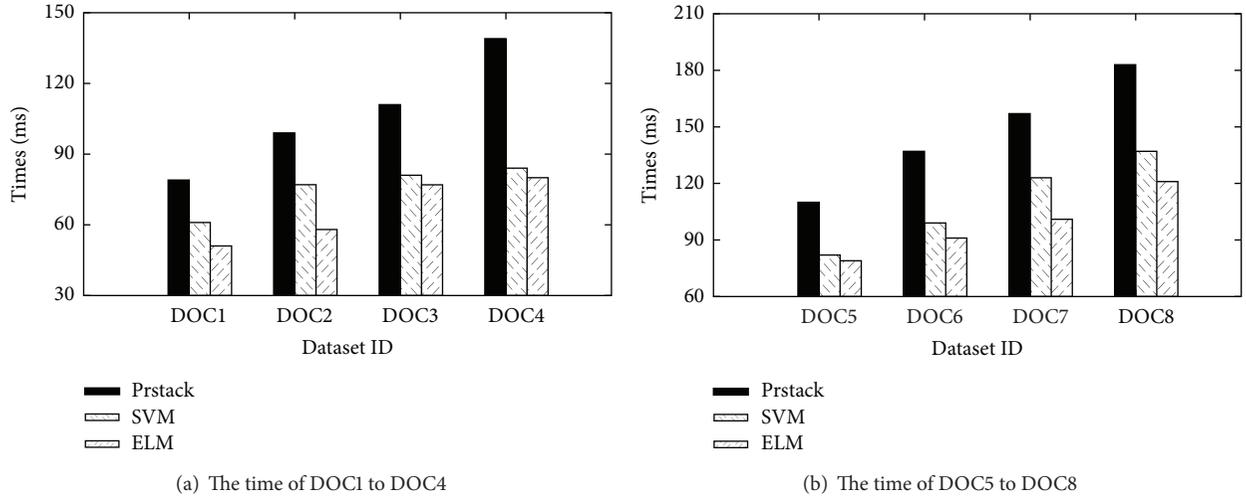


FIGURE 14: Vary query over DOC1 to DOC8.

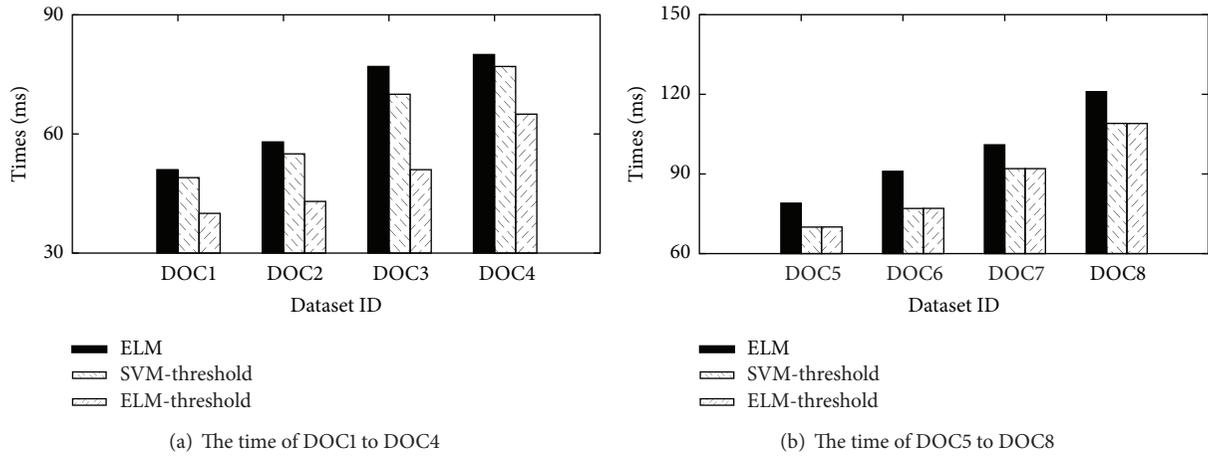


FIGURE 15: Vary query over DOC1 to DOC8.

TABLE 3: Keywords in probabilistic XML data.

| ID    | Name  | Keyword query               |
|-------|-------|-----------------------------|
| DOC 1 | XMARK | United States, Gredit       |
| DOC 2 | XMARK | United States, ship         |
| DOC 3 | XMARK | Ship, Credit, Alexas        |
| DOC 4 | XMARK | United States, ship, Gredit |
| DOC 5 | DBLP  | XML, Keyword                |
| DOC 6 | DBLP  | Keyword, query              |
| DOC 7 | DBLP  | XML, Keyword, probabilistic |
| DOC 8 | DBLP  | Keyword, query, XML         |

about keyword search in probabilistic XML data. The first situation is using the method in [17] to retrieve the SLCA nodes, and the second situation is using SVM and ELM to classify nodes for the keyword search on  $p$ -document. The second situation classifies nodes by using SVM and ELM. The speed of classification is shown in Figure 14.

From Figure 14 we can see that ELM has advantages of speed compared with Prstack and SVM. Prstack will compute all the nodes probabilities of all the ancestor nodes of the keyword node and it will record all the situations of the node which contains keywords. ELM can classify nodes according to the code and keywords by retrieving all the nodes once. So, the algorithm has high speed by using ELM to classify compared with SVM.

Next, we compare the query times of two situations about keyword search in probabilistic XML data. The first situation is using ELM to classify nodes for the keyword search on  $p$ -document, and the second situation is adding the probability threshold to classify nodes for the keyword search on  $p$ -document based on SVM. The third situation is adding the probability threshold to classify nodes for the keyword search on  $p$ -document based on ELM. The speed of classification is shown in Figure 15.

From Figure 15, we can see that the probability threshold has advantages. If the threshold is set to 0.4, from the results, we can see that compared with ELM algorithm,

the algorithm SVM-threshold and ELM-threshold which adds the probability threshold can also improve the time efficiency. ELM-threshold has advantages of speed compared with SVM-threshold. The pruning algorithm can be more than 1.3 times faster than the first algorithm. After adding probability threshold, the exist probability is less than probability threshold will be deleted; the probability will be also deleted when the probability is less than probability threshold on the process of computing probabilities.

## 7. Related Work

Recently, keyword search has been studied extensively in traditional XML data. For a keyword query and an XML tree, most of related work took SLCA and ELCA as the results to be returned. XRANK [1] developed stack-based algorithm to compute SLCA. Reference [2] introduced two algorithms, they are the Indexed Lookup Eager algorithm when the keywords appear with significantly different frequencies and the Scan Eager algorithm when the keywords have similar frequencies. Reference [3] designed an MS approach to compute SLCA for keyword queries in multiple ways. Reference [4] took set intersection to compute SLCA. Reference [18] proposed the Indexed Stack algorithm to find ELCA. The probabilistic XML model has been studied recently. In [6], they first introduced a probabilistic XML model with the probabilistic types IND and MUX. IND means independant and MUX means mutually exclusive. Reference [5] summarized and extended the probabilistic XML models previously proposed; the expressiveness and tractability of queries on different models are discussed with the consideration of IND and MUX. Reference [17] addressed the problem of keyword search in probabilistic XML data and computed SLCA by scanning the keyword inverted lists once. Different from all the above work, we adopt set intersection to compute SLCA in probabilistic XML data.

## 8. Conclusions

In this paper, we have addressed the problem of keyword search in a general probabilistic XML data. And we adopt probabilistic XML model  $\text{PrXML}^{\{\text{ind}, \text{mux}\}}$ . Given a probabilistic XML tree  $T$ , a set of keywords, and a probability threshold, we have discussed the challenges to find SLCA results with the probability which is more than probability threshold. Our algorithm has been proposed to compute the SLCA probabilities without generating possible worlds. We adopt set intersection to compute SLCA based on ELM. This paper uses ELM to classify nodes according to keyword search on probabilistic XML data. Keyword search on probabilistic XML data has received much attention in the literature. Finding efficient query processing method for keyword search on probabilistic XML data is an important topic in this area. In this paper, SLCA is selected as the results. Classification for nodes is important among all the operations. ELM can increase retrieval speed for the classification. So, ELM can support keyword search on probabilistic XML data. The experiments have demonstrated efficiency of our algorithms.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

Yue Zhao, Ye Yuan, and Guoren Wang were supported by the NSFC (Grant nos. 61025007, 61328202, and 61100024), National Basic Research Program of China p(973, Grant no. 2011CB302200-G), National High Technology Research and Development 863 Program of China (Grant no. 2012AA011004), and the Fundamental Research Funds for the Central Universities (Grant no. N130504006).

## References

- [1] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK: ranked keyword search over XML documents," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 16–27, 2003.
- [2] Y. Xu and Y. Papakonstantinou, "Efficient keyword search for smallest LCAs in XML databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*, pp. 527–538, June 2005.
- [3] C. Sun, C.-Y. Chan, and A. K. Goenka, "Multiway SLCA-based keyword search in XML data," in *Proceedings of the 16th International World Wide Web Conference (WWW '07)*, pp. 1043–1052, Alberta, Canada, May 2007.
- [4] J. Zhou, Z. Bao, W. Wang et al., "Fast SLCA and ELCA computation for XML keyword queries based on set intersection," in *Proceedings of the 28th International Conference on Data Engineering (ICDE '12)*, pp. 905–916, IEEE, Washington, DC, USA, April 2012.
- [5] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv, "Query efficiency in probabilistic XML models," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*, pp. 701–714, June 2008.
- [6] A. Nierman and H. V. Jagadish, "ProTDB: probabilistic data in XML," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*, pp. 646–657, 2002.
- [7] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, Hangzhou, China, June 2014.
- [8] J. W. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [9] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003.
- [10] G.-B. Huang and C.-K. Siew, "Extreme learning machine with randomly assigned RBF kernels," *International Journal of Information Technology*, vol. 11, pp. 16–24, 2005.
- [11] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.
- [12] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16-18, pp. 3056–3062, 2007.

- [13] J.-G. Taylor, "Cognitive computation," *Cognitive Computation*, vol. 1, no. 1, pp. 4–16, 2009.
- [14] M. Wöllmer, F. Eyben, A. Graves, B. Schuller, and G. Rigoll, "Bidirectional lstm networks for context-sensitive keyword detection in a cognitive virtual agent framework," *Cognitive Computation*, vol. 2, no. 3, pp. 180–190, 2010.
- [15] P. K. Mital, T. J. Smith, R. L. Hill, and J. M. Henderson, "Clustering of gaze during dynamic scene viewing is predicted by motion," *Cognitive Computation*, vol. 3, no. 1, pp. 5–24, 2011.
- [16] E. Cambria and A. Hussain, *Sentic Computing: Techniques, Tools, and Applications*, 2012.
- [17] J. Li, C. Liu, R. Zhou, and W. Wang, "Top-k keyword search over probabilistic XML data," in *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE '11)*, pp. 673–684, IEEE, Hannover, Germany, April 2011.
- [18] Y. Xu and Y. Papakonstantinou, "Efficient lca based keyword search in xml data," in *Proceedings of the 11th International Conference on Extending Database Technology (EDBT '08)*, 2008.

## Research Article

# Quasilinear Extreme Learning Machine Model Based Internal Model Control for Nonlinear Process

Dazi Li, Qianwen Xie, and Qibing Jin

*Institute of Automation, Beijing University of Chemical Technology, No. 15 East Road of the North 3rd Ring-Road, Chao Yang District, Beijing 100029, China*

Correspondence should be addressed to Dazi Li; [lidz@mail.buct.edu.cn](mailto:lidz@mail.buct.edu.cn)

Received 17 August 2014; Revised 21 October 2014; Accepted 22 October 2014

Academic Editor: Jiuwen Cao

Copyright © 2015 Dazi Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new strategy for internal model control (IMC) is proposed using a regression algorithm of quasilinear model with extreme learning machine (QL-ELM). Aimed at the chemical process with nonlinearity, the learning process of the internal model and inverse model is derived. The proposed QL-ELM is constructed as a linear ARX model with a complicated nonlinear coefficient. It shows some good approximation ability and fast convergence. The complicated coefficients are separated into two parts. The linear part is determined by recursive least square (RLS), while the nonlinear part is identified through extreme learning machine. The parameters of linear part and the output weights of ELM are estimated iteratively. The proposed internal model control is applied to CSTR process. The effectiveness and accuracy of the proposed method are extensively verified through numerical results.

## 1. Introduction

Internal model control is to design control strategy based on a kind of mathematical model of the process. Because of its obvious dynamic and static performance, as well as simple structure and strong robustness, internal model control plays an increasingly significant effect in control area [1, 2]. Two crucial problems in the inverse system approach are identification of plant model and determination of controller settings. For a complex nonlinear system, it is difficult to obtain an accurate internal model and its inverse model. In recent years, much effort has been devoted to nonlinear system modeling based on artificial neural networks (NNs) and support vector machine (SVM) [3–5]. It is widely applied to use the solution of trained inverse model as a nonlinear controller [6]. However, the disadvantages of the dynamic gradient method lie in its long training time, minor update of weights, and high probability of training failure. SVM method based on standard optimization often suffers from parameter adjustment difficulties. Moreover, the update information based on errors in internal model and inverse model also leads to decrease of the control performance [7].

To deal with the above problems, extreme learning machine (ELM) proposed by Huang et al. [8–10] shows great

advantages. Its simplified neural network structure makes the learning speed fast. A smaller training error can be obtained via a canonical equation. The advantage of ELM is its low computational effort and high generalization ability. Therefore, ELM has been successfully applied in many areas, such as classification of EEG signals and protein sequence [11], building regression model [12, 13], and fault diagnosis [14, 15].

For the nonlinear modeling, the key point is to find a suitable model structure. Volterra model is a kind of crucial nonlinear system model [16]. It provides an elaborate mathematical description for a great many of nonlinear systems. Recently, some researchers proposed the proof of inverse theory for IMC based on Volterra model [17]. However, the obvious shortcoming that limits its application is its high complexity in the identification of kernel function.

In recent years, some block-oriented models have been proposed and applied widely, such as Wiener model and Hammerstein model [18–21] which consist of a static nonlinear function and a linear dynamic subsystem. Both of them are of simple structures and can be used to identify some highly nonlinear process, such as pH neutralization process [22] and fermentation process [23]. However, sometimes it is difficult to separate the system concerned into a linear

dynamic block and a memoryless nonlinear one. Another class of methods based on local linearization of the structure, combining the nonlinear nonparametric models with some conventional statistical models, has achieved some great results. McLoone et al. [24] proposed an off-line hybrid training algorithm for feed-forward neural networks. Peng et al. [25, 26] proposed hybrid pseudolinear RBF-AR, RBF-ARX models. A cascaded structure of the ARX-NN model is proposed by Hu et al. [27]. The idea of these two different classes of methods is to separate the linear and nonlinear identification, so as to facilitate the inverse computation.

However, these models show high nonlinear characteristics, which are difficult to analysis in theory, without exploiting some good linearity properties. It is well known that simple structure, such as ARX model, has a lot of advantages in modeling. Firstly, its linear properties will significantly simplify the parameter estimation. Secondly, it is convenient to deduce regression predictor. Furthermore, linearity structure is also convenient for control design as well as the control law derivation. A good representation cannot only approximate the nonlinear function accurately, but also simplify the identification process. Hu et al. [28, 29] proposed a quasilinear model constructed by a linear structure using a quasilinear ARX model for nonlinear process mapping. From a macrostandpoint, the model can be seen as a linear structure which is a redundant for the regression ability. Its complex coefficients reflect the nonlinearity of the system. The model has a great flexibility to deal with the system nonlinearity.

Inspired by this kind of quasilinear ARX model as well as the thought of separate identification, the motivation of this paper is intended to propose a class of quasilinear ELM model, which can be separated into a linear part and a nonlinear kernel part. It cannot only identify ordinary nonlinear system, but also simplify the identification process via separating the model complexity. In this paper, a novel internal model control based on quasilinear-ELM (QL-ELM) structure is proposed for CSTR system. Taking advantage of separate identification, the quasilinear model consists of a linear part and a nonlinear kernel part. The parameters of nonlinear part are estimated by ELM, which increases the flexibility of the model. The linear parameters are estimated by using the RLS method. A recursive algorithm is conducted to estimate the parameters in both parts. Moreover, QL-ELM is used to set up the internal and inverse model of nonlinear CSTR systems. Through the establishment of the inverse model, the control action is obtained to achieve fixed-point control and tracking control of concentration. Taking the advantage of its characteristics of high modeling accuracy and less human interference, the closed-loop system control is more stable and has less steady-state deviation. Simulation results demonstrate the dynamic performance and tracking ability of the proposed QL-ELM based IMC strategy.

This paper is organized in six sections. Following the introduction, the traditional extreme learning machine is illustrated in Section 2. In Section 3 the algorithm of QL-ELM is presented. IMC with QL-ELM is described in Section 4. To show the applicability of the proposed method, simulations

results for CSTR are presented in Section 5. Finally, the conclusion is presented in Section 6.

## 2. Extreme Learning Machine: Basic Principles

For the input nodes  $x_i = [x_{i1}, x_{i2}, \dots, x_{im}]$  and output nodes  $y_i = [y_{i1}, y_{i2}, \dots, y_{im}]$  ( $i \in (1, 2, \dots, N)$ ) the single-hidden layer feed-forward neural networks (SLFNs) with  $M$  hidden nodes and activation function  $g(x)$  can be expressed as

$$\sum_{i=1}^M \beta_i g(w_i \cdot x_i + b_i) = y_i, \quad (1)$$

where  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the vector of weights between hidden layer and the output nodes.  $w_i = [w_{i1}, w_{i2}, \dots, w_{im}]^T$  is the vector of weights between input vectors and hidden layer. In addition,  $b_i$  is the bias of the  $i$ th hidden node. ELM with wide types of activation functions  $g(x)$  can get high regression accuracy. Unlike other traditional implementations, the input weights and biases are randomly chosen in extreme learning machine. The output of the hidden layer is written as a matrix  $H$ , and (1) can be rewritten as

$$H\beta = Y, \quad (2)$$

where

$$H = \begin{bmatrix} g(w_1 x_1 + b_1) & \cdots & g(w_M x_1 + b_M) \\ \vdots & \vdots & \vdots \\ g(w_1 x_N + b_1) & \cdots & g(w_M x_N + b_M) \end{bmatrix}_{N \times M}. \quad (3)$$

With the theorems proposed in [8, 9], the input weights  $w_i$  and the hidden layer biases  $b_i$  are randomly generated without further tuning. It is the main idea of the ELM that training problem is simplified to find a least square solution. According to the Moore-Penrose generalized inverse theory, the output can be calculated by using the following equation:

$$\hat{\beta} = H^\dagger Y. \quad (4)$$

It must be the smallest norm solution among all the solutions. The one step algorithm can produce best generalization performance and learn much faster than traditional learning algorithms. It can also avoid local optimum.

## 3. The Quasilinear ELM Model Treatment

A quasilinear ELM model can be seen as a SLFN embedded in the coefficients of a linear model. The feature of the quasilinear ELM model is that it has both good approximation abilities and easy-to-use properties. For a nonlinear SISO system described by

$$y(t) = f(X^T(t)) + e(t), \quad (5)$$

where  $X(t) = [y(t-1), y(t-2), \dots, y(t-n_a), u(t-1), \dots, u(t-n_b)]$ .  $X(t)$  is the regression vector,  $n_a, n_b$  are

the order of the system.  $n_a + n_b = d$ ,  $X(t) \in R^d$ .  $e(t)$  is a stochastic noise with zero-mean.

Assume that  $f(\cdot)$  is continuous and differentiable at a small region around  $X(t) = 0$ . By using Taylor equation [30],  $f(X^T(t))$  can be further expanded as

$$y(t) = f(0) + f'(0)X(t) + \frac{1}{2}X^T(t)f''(0)X(t) + \dots + \delta_n. \quad (6)$$

Set  $y_0 = f(0)$ ;  $\Theta(X(t))^T = (f'(0) + (1/2)X^T(t)f''(0) + \dots)$ . Picking up common factor  $X(t)$  from (6), then the following can be obtained:

$$\begin{aligned} y(t) &= y_0 + X^T(t)\Theta(X(t)) + e(t) \\ &= X^T(t)(\theta + \Theta(X(t))) + e(t), \end{aligned} \quad (7)$$

where  $(f'(0) + (1/2)X^T(t)f''(0) + \dots + \delta_n)^T = [a_{1,t}, \dots, a_{n,t}, b_{0,t}, \dots, b_{m-1,t}]^T$ . It can be seen as the coefficient of nonlinear function  $f(\cdot)$ .

The quasilinear model has a linear structure ARX model with a functional coefficient  $\Theta(X(t))$ . It can be separated into a nonlinear part  $y_N(t)$  and a linear part  $y_L(t)$  described as

$$y(t) = y_L(t) + y_N(t) + e(t) \quad (8)$$

$$y_L(t) = X^T\theta \quad (9)$$

$$y_N(t) = X^T(t)\Theta(X(t)), \quad (10)$$

where  $\Theta = [a_{1,t}, \dots, a_{n,t}, b_{0,t}, \dots, b_{m-1,t}]^T$ .

For case of near linear system, nonlinear part is the supplement for nonlinear feature, so good regression results can be achieved. For case of the nonlinear system, nonlinear network as an interpolated coefficient  $\Theta(X(t))$  can be used to expend the regression space. Equation (10) can be seen as the linear form with a nonlinear coefficient  $\Theta(X(t))$ , which is actually a problem of function approximation from a multidimensional input space  $X$  into a one-dimensional scalar space  $\Theta(X(t))$ . Using ELM to estimate nonlinear part parameters will be more convenient and concise. Replacing  $\Theta$  by ELM, the model in (7) can be rewritten as

$$y(t) = X^T(t) \sum_{i=1}^M N(X(t), \Omega) + e(t), \quad (11)$$

where  $N(X(t), \Omega) = W_2\Gamma(W_1X(t) + B) + \theta$ ; then the quasilinear ELM model can be further expressed as

$$y(t) = X^T(t) \sum_{i=1}^M W_i^2\Gamma(W_i^1X(t) + B) + X^T\theta + b + e(t), \quad (12)$$

where the activation function  $\Gamma(\cdot)$  is chosen as  $\Gamma(x) = 1/(1 + e^{-x})$ . The whole identification process based on QL-ELM is described in Figure 1, where  $T(t) = W_2\Gamma(W_1X(t) + B)$ ,  $n_a$  and  $n_b$  are orders of the input and output,  $W_1$  and  $W_2$

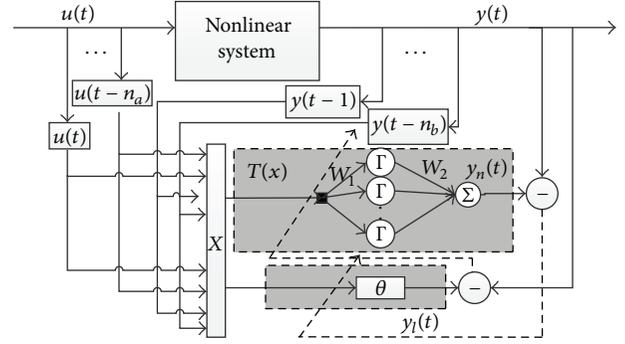


FIGURE 1: The process identification based on QL-ELM.

are weight matrices of the input and output layer,  $B$  is bias vector of hidden nodes, and  $\theta$  is the parameter of linear part and also can be seen as the bias vector of output nodes. The parameters of two submodels are updated during each iterative process until the ultimate goal to make the error between the output of actual model and the QL-ELM model minimized. The deviation between  $y(t)$  and  $y_L(t)$  is used to update the nonlinear part through ELM learning. The deviation between  $y(t)$  and  $y_N(t)$  is used to update the linear part through recursive least squares.

The whole process is to make the error between actual output and model output minimized. In this paper a hierarchical iterative algorithm is considered for quasilinear model.

For linear part, at every iteration, the following RLS is used to minimize the sum of squared residuals  $e = \|y_l - \hat{y}_l\|^2 = \|y_l - X^T\hat{\theta}\|^2$  avoiding the problems of local optimal and overfitting.

For nonlinear part, weights of input layer and biases are fixed and the training error  $e = \|HW_2 - T\|^2$  is minimized through ELM learning. Then the weights of output layer are calculated as  $W_2 = H^T T$ . The linear parameter  $\theta$  also can be regarded as noise. ELM method has capability of interference suppression and rapidity. Because the linear form of the model disperses the complexity of the nonlinear process, the computation of nonlinearity estimation can be simplified at every iteration. It means that less hidden nodes ( $M$ ) are required to avoid the overfitting problem in some extent. Using the QL-ELM model to identify the reversible model system can improve the identification accuracy and system performance.

#### 4. IMC with QL-ELM

In the nonlinear IMC, the nonlinear model and its inverse play an important role. In this study, QL-ELM is employed as both internal model and inverse model controller. The basic structure of QL-ELM based IMC is described as block diagram of Figure 2. There are four parts for the unknown nonlinear discrete control systems.  $G_p$  is the nonlinear plant; QL-ELM model is employed as inverse model controller ( $G_{IMC}$ ) and internal model ( $G_M$ ). In particular, the additional filter cannot only increase the physical realization of the controller, but also improve the robustness of the system.

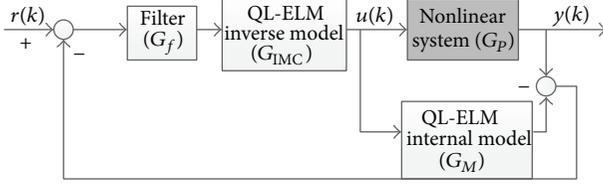


FIGURE 2: Structure of IMC system based on QL-ELM.

It can effectively solve problems caused by model mismatch [31].

*4.1. Establishment of Internal Model.* For a nonlinear plant described by

$$y(k+1) = f[y(k), y(k-1), \dots, y(k-n_a), u(k), u(k-1), \dots, u(k-n_b)], \quad (13)$$

where  $n_a$  and  $n_b$  is the order of output and input vector. The input vector  $x(k) = [y(k), y(k-1), \dots, y(k-n_a), u(k), u(k-1), \dots, u(k-n_b)]^T$  and output vector  $y(k+1)$  are used as samples to set up the internal model via QL-ELM.  $\{y(k+1), x(k)\}$ ,  $k \in 1, \dots, N-1$  is the training set. The learning of QL-ELM is implemented by the following steps.

*Step 1 (initialization).* Choose the order of the regression vector  $n_a, n_b$ . Set  $\theta$  to zero and the number of nodes in hidden layer  $M$  and nonlinear parameters  $W_1, W_2, B$  to some small values randomly. The number of iteration is set to  $n = 1$ .

*Step 2.* Update the linear part using deviation  $y_1(k) = y(k) - y_n(k)$  and estimate  $\theta_L$  using (9).

*Step 3.* Update the nonlinear part using  $y_n(k) = y(k) - y_1(k)$  and estimate  $\theta_N(W_2, k)$  using (10).

*Step 4.* Turn to Step 2, and set  $n = n + 1$  until the training error  $e = \|Y - \hat{Y}\|^2$  reaches minimum.

*4.2. Establishment of Inverse Model.* The controller of IMC is the inverse model of the nonlinear process which is equivalent to finding the inverse of the system at given frequencies. Therefore the reversibility of the model must be considered in advance.

**Theorem 1.** *For the above nonlinear system, if the  $x(k) = [y(k+1), y(k), \dots, y(k-n_a), u(k-1), \dots, u(k-n_b)]$  is monotone function to  $u(k)$ , the system is reversible. Or for any given two inputs  $u_1(k), u_2(k)$  if  $f(y(k), \dots, y(k-n_a), u_1(k), \dots, u_1(k-n_b)) \neq f(y(k), \dots, y(k-n_a), u_2(k), \dots, u_2(k-n_b))$  holds, the system is reversible [7].*

Assume a SISO nonlinear process is described in (13); the inverse model is established as

$$u(k) = g[y(k+1), y(k), \dots, y(k-n_a), u(k-1), \dots, u(k-n_b)], \quad (14)$$

TABLE 1: Physical parameters for the CSTR model.

| Parameters | Nomenclature              | Value |
|------------|---------------------------|-------|
| $B$        | Heat of reaction          | 8     |
| $\delta$   | Heat transfer coefficient | 0.3   |
| $Da$       | Damökhler number          | 0.072 |
| $\varphi$  | Activated energy          | 20    |

where  $g(\cdot)$  is nonlinear function of inverse model. According to Theorem 1 the process of (14) is monotone and reversible. Training the QL-ELM can establish the inverse model of system. The input and output vectors are  $x(k) = [y(k+1), y(k), \dots, y(k-n_a), u(k-1), \dots, u(k-n_b)]$  and  $u(k)$ , respectively. Because the value of  $y(k+1)$  is unknown, the output of filter  $r(k)$  replaces the value in the above formula. Training process of inverse model is the same as the internal model.

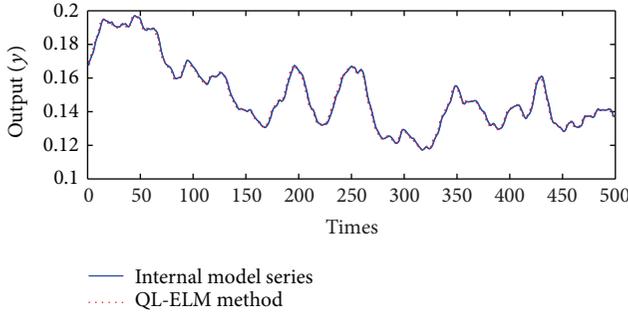
## 5. Numerical Results

A typical representative of nonlinear system in chemical processes is CSTR system. The system has multiple equilibrium points (stable and unstable ones). Its dynamical behavior exhibits some complex features depending on system parameters. In this study, the dynamic behavior is described by the following differential equations [32, 33]:

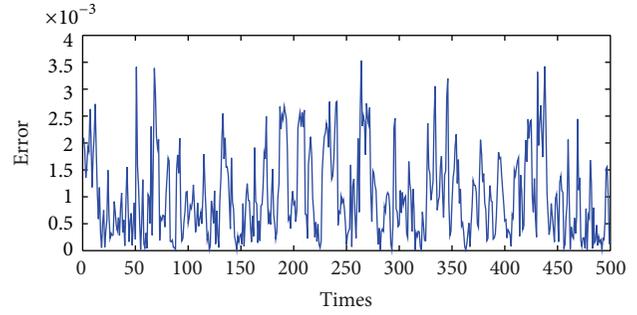
$$\begin{aligned} \dot{x}_1 &= -x_1 + Da(1-x_1) \exp\left(\frac{x_2}{1+x_2/\varphi}\right) + d_1, \\ \dot{x}_2 &= -(1+\delta)x_2 + B \cdot Da(1-x_2) \exp\left(\frac{x_2}{1+x_2/\varphi}\right) \\ &\quad + \delta \cdot u + d_2, \\ y &= x_1, \end{aligned} \quad (15)$$

where  $x_1$  and  $x_2$  represent the dimensionless reactant concentration and reactor temperature, respectively;  $d_1$  and  $d_2$  denote the system disturbances. The control action  $u$  is the cooling jacket temperature. The model parameters are shown in Table 1. The model has three equilibrium points, where  $(x_1, x_2) = (0.144, 0.886)$  and  $(x_1, x_2) = (0.765, 4.705)$  are stable points and  $(x_1, x_2) = (0.445, 2.74)$  is unstable point. The reactant concentration  $x_1$  is chosen as the controlled variable. The resulting control problem is nonlinear. Therefore, training of the models has to be restricted to a region where inverse mapping is unique to ensure the reversibility.

The initial condition is set as  $[x_1(0), x_2(0)] = [-0.1, 0.1]$ , and  $u(t) = 2 * \text{rand}n(1)$ . The fourth-order Runge-Kutt algorithm is used to calculate this model with the integral step size  $\Delta t = 0.1$ . The number of hidden nodes  $M$  is 80. In the simulation, modelling error caused by the lack of the training sample will lead to the residual of the control system. Therefore, some steady-state data around the stable point are added as training samples. Results of model based on QL-ELM are compared with those of ELM, SVM, and QL-SVM methods. In detail, the number of hidden nodes in QL-ELM is reduced by 40. The optimal parameters of SVM and QL-SVM

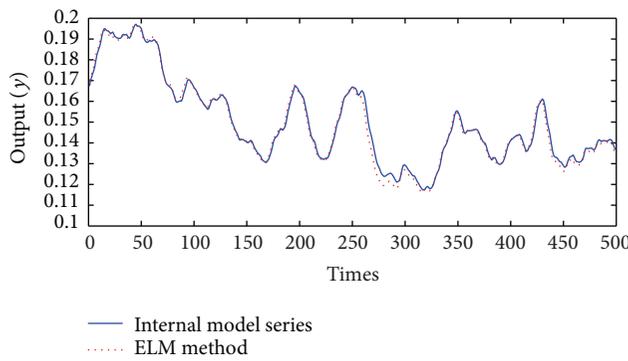


(a) QL-ELM method output and the actual output of internal model

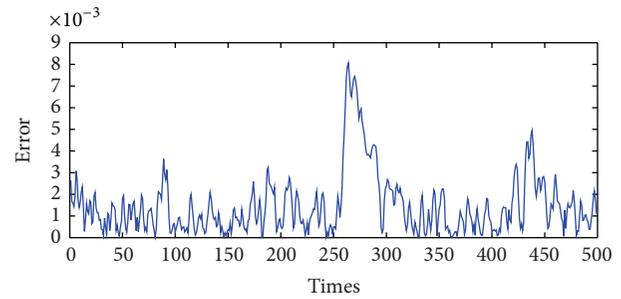


(b) Error of QL-ELM internal model

FIGURE 3: Identification results of internal model using QL-ELM.



(a) ELM method output and the actual output of internal model



(b) Error of ELM internal model

FIGURE 4: Identification results of internal model using ELM.

with RBF kernel are selected using the cross-validation. For the SVM method, the scale parameters in internal model are set as the penalty factor  $C = 500$ , the variance in RBF kernel function  $\delta = 0.01$ , and the epsilon in loss function of SVR  $e = 0.001$ . In the inverse model the parameters are  $C = 10000$ ,  $\delta = 0.01$ , and  $e = 7$ . For the QL-SVM method the parameters in internal model and inverse model are  $C = 100$ ,  $\delta = 0.001$ ,  $e = 0.001$  and  $C = 3000$ ,  $\delta = 0.001$ ,  $e = 2$ , respectively.

For the internal model, the order of internal model is set as  $n_a = n_b = 1$ ; 2000 groups of samples are chosen as the training data and 500 groups of samples are chosen as test data.

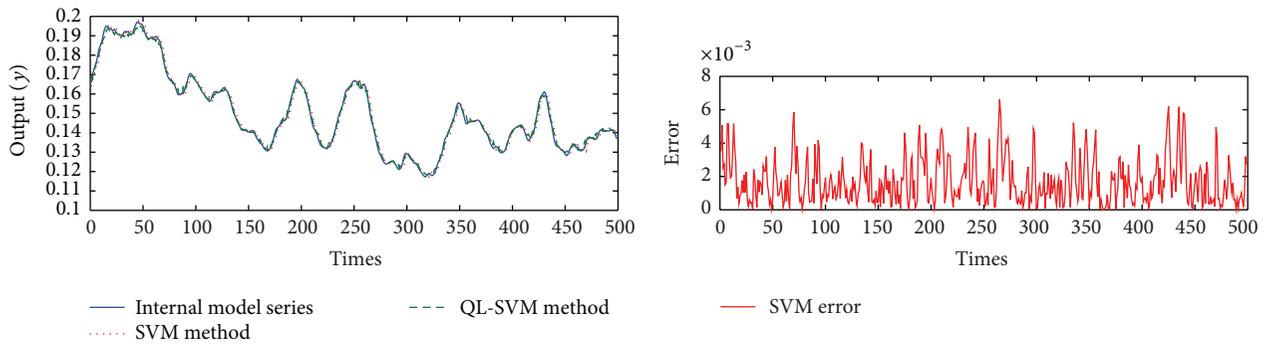
For the inverse model  $x(k) = [x_1(k-1), x_2(k-1), x_1(k), x_2(k), x_1(k+1), u(k-1)]$ ;  $y(k) = u(k)$ ; 2000 groups of data are chosen for inverse model training so as to get the controller, and the remaining 500 groups of data are chosen for the inverse model test.

The performance of modelling is measured by the root mean square error (RMSE), and the indicator can be expressed by

$$P = \sqrt{\frac{1}{N} \sum_{k=1}^N (y(k) - y(k | \theta))^2}. \quad (16)$$

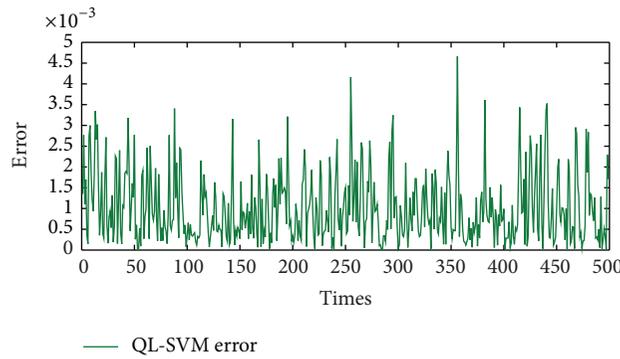
Identification results of the internal model and inverse model with the QL-ELM and its corresponding error are

shown in Figures 3 and 7. In addition, results and corresponding modeling error of the comparative ELM method are shown in the Figures 4 and 8, respectively. Validation results of SVM and QL-SVM based internal model are shown in Figure 5(a) and their corresponding errors are shown in Figures 5(b) and 5(c). The enlarged detail of Figure 5(a) is shown in Figure 6. Similarly, validation results of SVM and QL-SVM based inverse model are shown in Figure 9(a) and their corresponding errors are shown in Figures 9(b) and 9(c). The enlarged detail of Figure 9(a) is shown in Figure 10. Obviously, both in the internal model and in inverse model identification, errors of the proposed method are smaller than other methods. Quasilinear method can get a better generalization performance compared with normal method. In addition, ELM method reduces the regression error. Combining with both advantages, the proposed QL-ELM method provides high precision and better generalization. To illustrate the effectiveness of the proposed method, the measurable indicator of different identification methods are listed in Table 2. In the comparisons of the time performance, it is worth mentioning that, for SVM, the time means once running time for SVM with the optimal parameter, without considering the time of parameters adjustment. It is obvious that QL-ELM could provide higher precision and less time consumption than other methods.



(a) Identification results of internal model using SVM and QL-SVM

(b) Test error of internal model using SVM



(c) Test error of internal model using QL-SVM

FIGURE 5: Comparison results of internal model using SVM and QL-SVM.

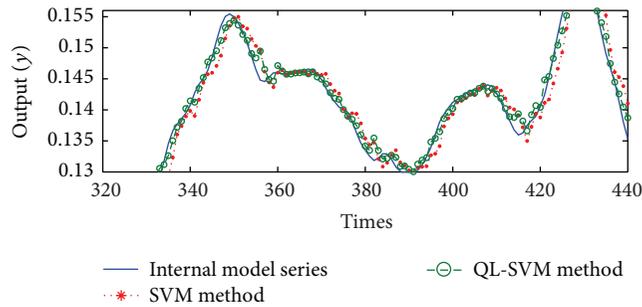
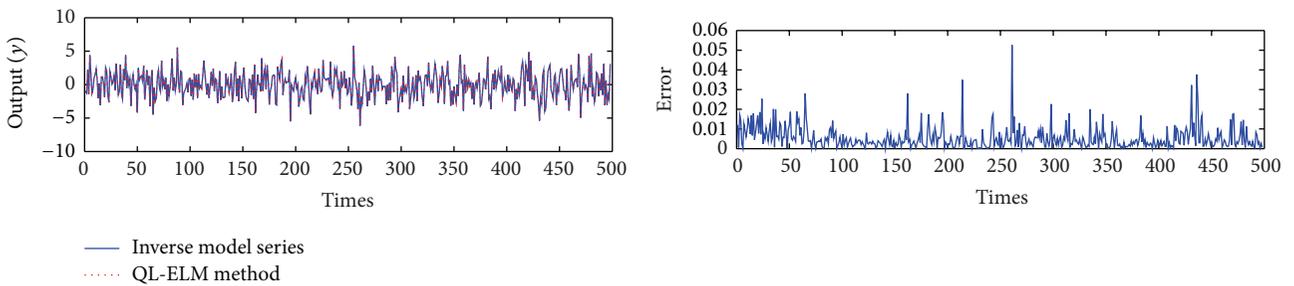


FIGURE 6: The details of Figure 5(a).



(a) QL-ELM method output and the actual output of inverse model

(b) Error of QL-ELM inverse model

FIGURE 7: Identification results of inverse model using QL-ELM.

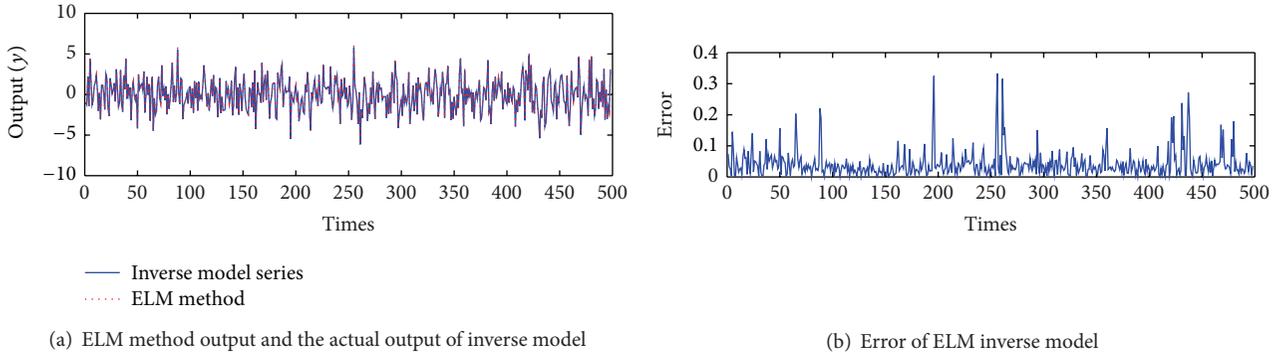


FIGURE 8: Identification results of inverse model using ELM.

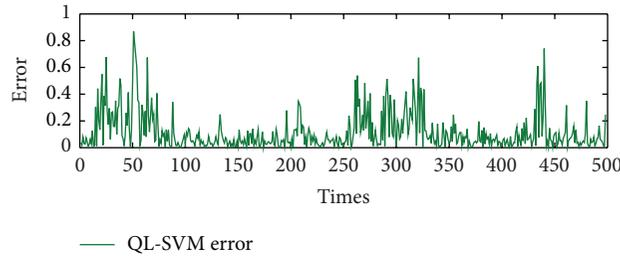
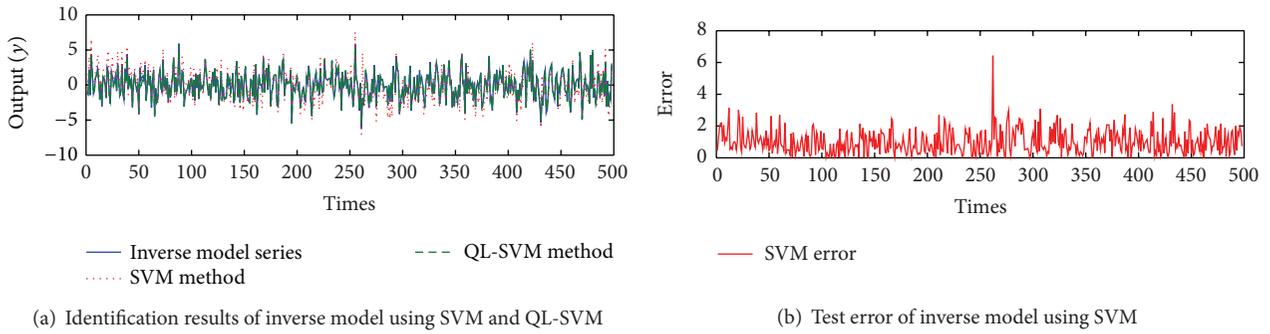


FIGURE 9: Comparison results of inverse model using SVM and QL-SVM.

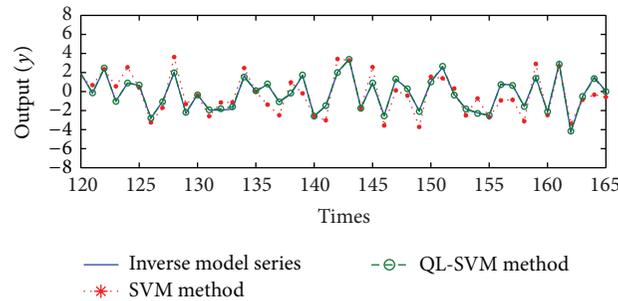
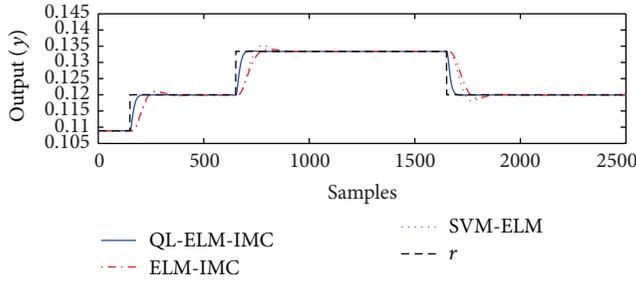


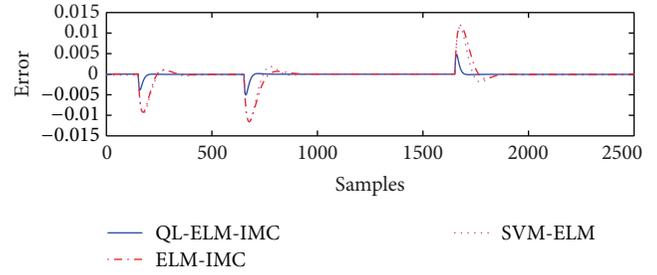
FIGURE 10: The details of Figure 9(a).

TABLE 2: Comparison of the identification error and time between different methods.

|                | QL-ELM |          | ELM    |          | SVM    |          | QL-SVM |          |
|----------------|--------|----------|--------|----------|--------|----------|--------|----------|
|                | RMSE   | TIMES    | RMSE   | TIMES    | RMSE   | TIMES    | RMSE   | TIMES    |
| Internal model | 0.0017 | 1.8929 s | 0.0026 | 2.1998 s | 0.0025 | 4.2639 s | 0.0019 | 3.7194 s |
| Inverse model  | 0.0122 | 1.8108   | 0.0582 | 2.0376 s | 1.2303 | 6.8584 s | 0.2659 | 6.4534 s |

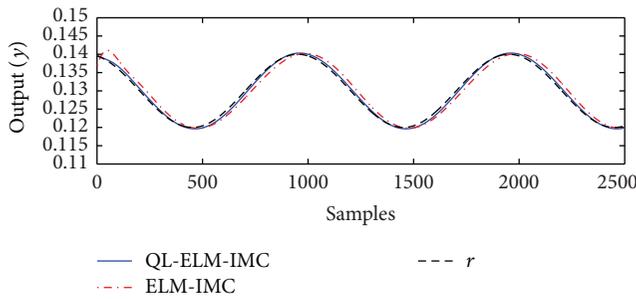


(a) Control output using IMC based on QL-ELM, ELM, and SVM

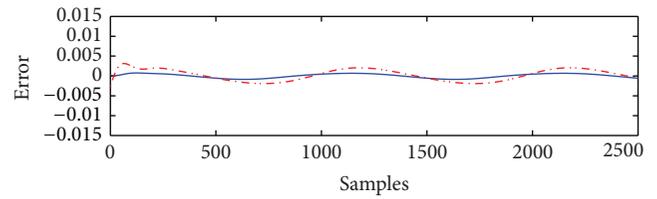


(b) Error of output using IMC based on QL-ELM, ELM, and SVM

FIGURE 11: Simulation results of set-point control.



(a) Control output using IMC based on QL-ELM, ELM



(b) Error of output using IMC based on QL-ELM, ELM

FIGURE 12: Simulation results of tracking control.

**5.1. Control under Set-Point Change.** The set-point is given as follows:  $r = 0.1088$  for  $k < 200$ ,  $r = 0.1200$  for  $200 < k < 700$ ,  $r = 0.1334$  for  $700 < k < 1700$ , and  $r = 0.1200$  for  $k < 2500$ . The initial condition of nonlinear CSTR process is designed as  $[x(1), x(2)] = [0, 0]$ , respectively. Besides, the first order filter is  $F = (1 - \beta)/(1 - \beta z^{-1})$ , where  $\beta = 0.9$ . Simulation results by IMC based on QL-ELM, ELM, and SVM are displayed in Figure 11. It can be easily seen that the proposed method has faster regulation time, smaller overshoot, and smaller steady-state error. Therefore, it is not difficult to conclude that the proposed method is superior to the ELM and SVM method in this control case.

**5.2. Tracking Control for a Sinusoidal Wave Input.** In this case, the nonlinear system output response to track a desired sinusoidal function is  $r(k) = 0.01 \sin(2\pi t/1000) + 0.13$ . The first order filter is  $F = (1 - \beta)/(1 - \beta z^{-1})$ , where  $\beta = 0.9$ . All the parameters settings for CSTR system are the same as Section 5.1. Comparisons of output for tracking control and error of output by the proposed method and ELM method are displayed in Figure 12. It is clearly seen that the system output can track the desired sinusoidal function perfectly. It can be concluded that the proposed method gives better tracking performance.

Although QL-ELM model reveals better approximation performance in all the above experiments, the proposed method still lacks versatility in extremely strong nonlinear process modelling. The shortage is caused by its essential linear structure.

## 6. Conclusion

Nonlinear control strategy by QL-ELM based IMC is proposed and tested on the concentrate control of SISO CSTR process. The internal model and inverse model controller are learned using QL-ELM regression algorithm to improve the modeling accuracy. The proposed QL-ELM method represents a great flexibility and simplicity via a quasilinear ARX model with ELM coefficient. It cannot only approximate continuous nonlinear function but also separate the nonlinear complexity with less hidden neurons in ELM. Taking advantage of the high accuracy of QL-ELM modeling as well as increased training samples around stable points, the steady-state error is deduced in IMC system. Simulation results reveal the superiority of the proposed method in good tracking ability and stability nonlinear process control.

## Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

## Acknowledgment

This research was supported by the National Natural Science Foundation of China (Grant no. 61273132, 61104098, 61473024).

## References

- [1] H. Deng and H.-X. Li, "A novel neural approximate inverse control for unknown nonlinear discrete dynamical systems," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 35, no. 1, pp. 115–123, 2005.
- [2] H. Yu, H. R. Karimi, and X. Zhu, "Research of smart car's speed control based on the internal model control," *Abstract and Applied Analysis*, vol. 2014, Article ID 274293, 5 pages, 2014.
- [3] I. Rivals and L. Personnaz, "Nonlinear internal model control using neural networks: application to processes with delay and design issues," *IEEE Transactions on Neural Networks*, vol. 11, no. 1, pp. 80–90, 2000.
- [4] Y.-N. Wang and X.-F. Yuan, "SVM approximate-based internal model control strategy," *Acta Automatica Sinica*, vol. 34, no. 2, pp. 172–179, 2008.
- [5] I. Rivals and L. Personnaz, "Nonlinear internal model control using neural networks: application to processes with delay and design issues," *IEEE Transactions on Neural Networks*, vol. 11, no. 1, pp. 80–90, 2000.
- [6] H.-X. Li and H. Deng, "An approximate internal model-based neural control for unknown nonlinear discrete processes," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 659–670, 2006.
- [7] Y. Huang and D. Wu, "Nonlinear internal model control with inverse model based on extreme learning machine," in *Proceedings of the International Conference on Electric Information and Control Engineering (ICEICE '11)*, pp. 2391–2395, Wuhan, China, April 2011.
- [8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
- [9] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.
- [10] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.
- [11] Y. Song and J. Zhang, "Automatic recognition of epileptic EEG patterns via Extreme Learning Machine and multiresolution feature extraction," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5477–5489, 2013.
- [12] Y. Chen, Z. Zhao, S. Wang, and Z. Chen, "Extreme learning machine-based device displacement free activity recognition model," *Soft Computing*, vol. 16, no. 9, pp. 1617–1625, 2012.
- [13] J. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1163–1168, 2014.
- [14] P. K. Wong, Z. Yang, C. M. Vong, and J. Zhong, "Real-time fault diagnosis for gas turbine generator systems using extreme learning machine," *Neurocomputing*, vol. 128, pp. 249–257, 2014.
- [15] G. Wang, Y. Zhao, and D. Wang, "A protein secondary structure prediction framework based on the Extreme Learning Machine," *Neurocomputing*, vol. 72, no. 1–3, pp. 262–268, 2008.
- [16] R. K. Pearson and B. A. Ogunnaike, *Identification and Control Using Volterra Models*, Springer, 2002.
- [17] K. T. Iskakov and Z. O. Oralkbekova, "Resolving power of algorithm for solving the coefficient inverse problem for the geoelectric equation," *Mathematical Problems in Engineering*, vol. 2014, Article ID 545689, 9 pages, 2014.
- [18] S. I. Biagiola and J. L. Figueroa, "Identification of uncertain MIMO Wiener and Hammerstein models," *Computers & Chemical Engineering*, vol. 35, no. 12, pp. 2867–2875, 2011.
- [19] Y. Tang, Z. Li, and X. Guan, "Identification of nonlinear system using extreme learning machine based Hammerstein model," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 9, pp. 3171–3183, 2014.
- [20] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of Hammerstein-Wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.
- [21] D. Wang and F. Ding, "Extended stochastic gradient identification algorithms for Hammerstein-Wiener ARMAX systems," *Computers & Mathematics with Applications*, vol. 56, no. 12, pp. 3157–3164, 2008.
- [22] J. G. Smith, S. Kamat, and K. P. Madhavan, "Modeling of pH process using wavenet based Hammerstein model," *Journal of Process Control*, vol. 17, no. 6, pp. 551–561, 2007.
- [23] L. Zhou, X. Li, and F. Pan, "Gradient-based iterative identification for MISO Wiener nonlinear systems: application to a glutamate fermentation process," *Applied Mathematics Letters*, vol. 26, no. 8, pp. 886–892, 2013.
- [24] S. McLoone, M. D. Brown, G. Irwin, and G. Lightbody, "A hybrid linear/nonlinear training algorithm for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 669–684, 1998.
- [25] H. Peng, T. Ozaki, V. Haggan-Ozaki, and Y. Toyoda, "A parameter optimization method for radial basis function type models," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 432–438, 2003.
- [26] H. Peng, T. Ozaki, Y. Toyoda et al., "RBF-ARX model-based nonlinear system modeling and predictive control with application to a NOx decomposition process," *Control Engineering Practice*, vol. 12, no. 2, pp. 191–203, 2004.
- [27] B. Hu, Z. Zhao, and J. Liang, "Multi-loop nonlinear internal model controller design under nonlinear dynamic PLS framework using ARX-neural network model," *Journal of Process Control*, vol. 22, no. 1, pp. 207–217, 2012.
- [28] J. Hu, K. Kumamaru, and K. Inoue, "A hybrid quasi-ARMAX modeling scheme for identification of nonlinear systems," *Transactions of the Society of Instrument and Control Engineers*, vol. 34, no. 8, pp. 977–985, 1998.
- [29] Y. Cheng and J. Hu, "Nonlinear system identification based on SVR with quasi-linear kernel," in *Proceedings of the Annual International Joint Conference on Neural Networks (IJCNN '12)*, pp. 1–8, June 2012.
- [30] G. Prasad, E. Swidenbank, and B. W. Hogg, "A local model networks based multivariable long-range predictive control strategy for thermal power plants," *Automatica*, vol. 34, no. 10, pp. 1185–1204, 1998.
- [31] Y. Zhang and Z. Zheng, "Based on inverse system of internal model control," in *Proceedings of the International Conference on Computer Application and System Modeling (ICCASM '10)*, pp. V14–19–V14–21, Taiyuan, China, October 2010.
- [32] C.-T. Chen and S.-T. Peng, "Intelligent process control using neural fuzzy techniques," *Journal of Process Control*, vol. 9, no. 6, pp. 493–503, 1999.
- [33] C.-T. Chen and S.-T. Peng, "Learning control of process systems with hard input constraints," *Journal of Process Control*, vol. 9, no. 2, pp. 151–160, 1999.

## Research Article

# Sample-Based Extreme Learning Machine with Missing Data

Hang Gao, Xin-Wang Liu, Yu-Xing Peng, and Song-Lei Jian

*Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha 410073, China*

Correspondence should be addressed to Hang Gao; [hanggao1821@163.com](mailto:hanggao1821@163.com)

Received 21 August 2014; Revised 8 November 2014; Accepted 10 November 2014

Academic Editor: Zhan-li Sun

Copyright © 2015 Hang Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Extreme learning machine (ELM) has been extensively studied in machine learning community during the last few decades due to its high efficiency and the unification of classification, regression, and so forth. Though bearing such merits, existing ELM algorithms cannot efficiently handle the issue of missing data, which is relatively common in practical applications. The problem of missing data is commonly handled by imputation (i.e., replacing missing values with substituted values according to available information). However, imputation methods are not always effective. In this paper, we propose a sample-based learning framework to address this issue. Based on this framework, we develop two sample-based ELM algorithms for classification and regression, respectively. Comprehensive experiments have been conducted in synthetic data sets, UCI benchmark data sets, and a real world fingerprint image data set. As indicated, without introducing extra computational complexity, the proposed algorithms do more accurate and stable learning than other state-of-the-art ones, especially in the case of higher missing ratio.

## 1. Introduction

Extreme learning machine (ELM) was proposed by Huang et al. in [1]. It works for the generalized single-hidden layer feedforward networks (SLFNs) [2]. Different from traditional tenet in neural network learning, hidden node parameters of ELM are randomly generated and do not need to be adjusted [3]. Therefore, it achieves fast learning speed as well as excellent generalization ability with least human intervention. As an open, scalable, and unified learning framework, ELM can be used in various kinds of learning tasks, for example, classification, regression, representation learning, and so forth. Its learning theory has become an active research topic in machine learning domain in recent years. On one hand, many improved variants have been developed. On another hand, ELM learning mechanism has been integrated into some well-known platforms and systems. The following is a brief review. The main objective of variants is mainly twofold: (1) achieving more accurate learning by introducing different regularization methodologies and (2) reducing computational complexity by simplifying ELM network architecture. Regularized ELM [4] considers heteroscedasticity in real applications and reduces the effect

of outliers in the data set. In [5], ridge regression, elastic net, and lasso methods are used to prune ELM network, which leads to compact architecture. Sparse Bayesian ELM [6] estimates the marginal likelihood of network outputs and automatically prunes most of the redundant hidden neurons. Localized error model based ELM utilizes principal component analysis to reduce the feature dimension and then selects the optimal architecture of the SLFN. OP-ELM uses LASSO regularization technique to rank the neurons of hidden layer and obtains a more parsimonious model. TROP-ELM [7] further improves OP-ELM by using a cascade regularization method based on LASSO and Tikhonov criteria. OP-ELM-ER-NCL [8] combines ensemble of regularization techniques with negative correlation penalty. TS-ELM [9] introduces a systematic two-stage mechanism to determine the network architecture. MK-ELM [10] considers multiple heterogeneous data sources and uses a multikernel strategy to optimize ELM. Moreover, ELM is modified according to different characteristics of data sets and applied in various applications. OS-ELM [11] learns the data one-by-one or chunk-by-chunk with fixed or varying chunk size. EOS-ELM [12] further improves OS-ELM's learning stability by adapting the node location, adjustment, and pruning methods. Weighted ELM

[13] assigns different weights to different samples according to users' needs, realizing cost sensitive learning. T2FELA [14] inherits the merits of ELM and randomly generates the parameters of the antecedents, successfully applying ELM in type 2 fuzzy logic system. In [15], ELM is integrated into MapReduce framework for large scale regression. All in all, those extensions and variants inherit the merits of ELM and are more suitable for specific application scenarios. However, missing data problem which is highly pervasive in real world tasks is rarely considered in ELM learning.

Missing data is a common situation where null value is stored for some samples in the data sets. This problem has complex patterns. It occurs by several practical reasons such as equipment malfunction in data sampling or transmission and noise value being deleted in data preprocessing [16]. Besides, there are some inherently missing caused by non-existing features. For example, an image sample may not contain all the predefined components. Standard ELM learning requires all samples in the data set to be complete and have the same dimensions. Obviously, it cannot directly handle the issue of missing data. Traditional approaches depend on the preprocessing for missing data before learning. Inevitably, those approaches introduce extra preprocessing overhead. Worse still, they may mistakenly omit some useful information and produce a certain amount of error information [17]. Therefore, learning precision can be seriously affected. In this paper, we propose a sample-based learning framework and develop a sample-based ELM classification algorithm and a sample-based ELM  $\epsilon$ -insensitive regression algorithm. In the proposed learning framework, missing data can be directly learned without any extra preprocessing.

The contributions of this paper are as follows. First, we analyze the limitations of state-of-the-art approaches for learning with missing data. Second, we propose a sample-based ELM learning framework for learning missing data. Third, we develop two sample-based ELM algorithms for classification and regression, respectively. Experiment results show that the proposed algorithms achieve more accurate classification and regression compared with traditional approaches, especially in the situation that missing ratio is relatively intensive.

## 2. ELM Learning

In Section 2.1, we review the basic concepts of ELM. Then, from general ELM optimization formula, we discuss several ELM variants and illustrate their characteristics. In Section 2.2, we introduce ELM  $\epsilon$ -insensitive regression and explain its advantage over standard ELM regression.

**2.1. Standard ELM.** From the viewpoint of neural network, ELM can be seen as generalized SLFNs. Figure 1 gives the network architecture. ELM randomly chooses hidden nodes and analytically determines the output weights of SLFNs. In theory, ELM can approximate any target continuous function and classify any disjoint regions [2]. Its interpolation capability and universal approximation capability have been

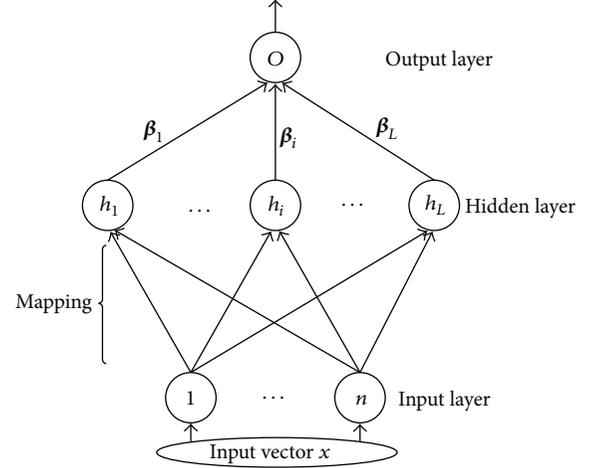


FIGURE 1: ELM's network architecture.

investigated by Huang et al. in [18, 19]. The corresponding output function is given as

$$f(\mathbf{x}) = \sum_{i=1}^L \beta_i * h(\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_i) \cdot \boldsymbol{\beta}, \quad (1)$$

where  $\boldsymbol{\beta}$  is the output weight vector and  $\mathbf{h}(\mathbf{x})$  is the random mapping function. Being superior to traditional learning algorithm, ELM tends to not only minimize the empirical risk but also minimize the structural risk. Its general optimization formula is as

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_p^{\sigma_1} + C \times \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_q^{\sigma_2}, \quad (2)$$

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{pmatrix} = \begin{pmatrix} h_1(x_1) & \cdots & h_L(x_1) \\ \vdots & \vdots & \vdots \\ h_L(x_N) & \cdots & h_L(x_N) \end{pmatrix}. \quad (3)$$

From the viewpoint of learning theory, ELM considers empirical risk as well as structural risk which can be observed in ELM's general optimization formula (2).  $\mathbf{H}$  is the random mapping matrix, and  $\mathbf{Y}$  is the training data targets. Evolved from (1), with different combinations of parameters and constraints in general formula, ELM derives many variants of different regularization methods. Additionally, kernel methods are used to enhance the learning ability especially for multiple data sources learning [20]. With competitive performance in generalization abilities, they are widely used in classification, regression, representation learning and clustering, and so forth. The following are some representative forms.

**Basic ELM.** There are two kinds of basic ELM. When  $C = +\infty$  (i.e., ELM only concerns empirical risk), the optimization objective equals  $\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_q^{\sigma_2}$ . In another extreme; when  $C = 0$  (i.e., ELM only concerns the structural risk), the optimization objective equals  $\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_p^{\sigma_1}$ . Obviously, basic ELM learning has high learning efficiency. But both of them take only one optimization objective into account, which can cause overfilling or underfitting.

*Inequality Optimization Constraints Based ELM.* With the parameter setting of  $\sigma_1 = 2$ ,  $\sigma_2 = 1$ ,  $p = 2$ ,  $q = 1$ , and inequality constraints, the general optimization formula can be written as (4), which is common in binary classification. Since this form is applied widely and has good sparsity, we use it as the base model of our extension for classification:

$$\begin{aligned} \min_{\boldsymbol{\beta}^{(i)}, \xi_i} \quad & \frac{1}{2} \|\boldsymbol{\beta}^{(i)}\|^2 + C \sum_{i=1}^n \xi_i, \\ \text{s.t.} \quad & \mathbf{y}_i * \left( \mathbf{h}(\mathbf{x}_i) \cdot \boldsymbol{\beta}^{(i)T} \right) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \quad (4)$$

Applying KKT conditions, (4) can be transformed into (5); then it can be solved in dual space:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{h}(x_i), \mathbf{h}(x_j) \rangle, \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n. \end{aligned} \quad (5)$$

*Equality Optimization Constraints Based ELM.* With the parameter setting of  $\sigma_1 = 2$ ,  $\sigma_2 = 1$ ,  $p = 2$ ,  $q = 1$ , and equality constraint, the general ELM optimization formula is equivalent to (6) which can be used in regression and classification:

$$\begin{aligned} \min_{\boldsymbol{\beta}^{(i)}, \xi_i} \quad & \frac{1}{2} \|\boldsymbol{\beta}^{(i)}\|^2 + C \sum_{i=1}^n \xi_i^2, \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_i) \cdot \boldsymbol{\beta}^{(i)T} = \mathbf{y}_i - \xi_i, \quad i = 1, 2, \dots, n. \end{aligned} \quad (6)$$

The corresponding KKT optimal conditions are shown in

$$\begin{aligned} \boldsymbol{\beta} &= \mathbf{H}\boldsymbol{\alpha}, \\ \alpha_i &= C\xi_i, \quad i = 1, 2, \dots, n, \\ \mathbf{h}(\mathbf{x}_i) \boldsymbol{\beta} - \mathbf{y}_i^T + \xi_i^T &= 0, \quad i = 1, 2, \dots, n. \end{aligned} \quad (7)$$

Further, the final output is given in

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \boldsymbol{\beta} = \mathbf{h}(\mathbf{x}) \mathbf{H}^T \left( \frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (8)$$

*2.2. ELM for  $\epsilon$ -Insensitive Regression.* For regression, ELM provides general model for standard setting. It achieves better predictive accuracy than traditional SLFNs [3]. In addition, many variants and extensions of ELM regression algorithms have been proposed. Inspired by Vapnik's epsilon insensitive loss function, [21] proposed  $\epsilon$ -insensitive ELM. Its optimization formula is as

$$\min_{\boldsymbol{\beta}} \left( \frac{1}{2} \times \|\boldsymbol{\beta}\|^2 + \frac{C}{2} \times \|\mathbf{h}(\mathbf{x}) \boldsymbol{\beta} - \mathbf{y}\|_{\epsilon}^2 \right), \quad (9)$$

where  $\epsilon$  is insensitive factor and the error loss function is calculated by

$$\|\mathbf{h}(\mathbf{x}) \boldsymbol{\beta} - \mathbf{y}\|_{\epsilon}^2 = \sum_{i=1}^n |f(x_i) - y_i|_{\epsilon}^2, \quad (10)$$

with  $|f(x_i) - y_i|_{\epsilon} = \max\{0, |f(x_i) - y_i| - \epsilon\}$ .

Compared with conventional ELM regression, ELM with  $\epsilon$ -insensitive loss function uses margin  $\epsilon$  to measure the empirical risk. It controls the sparsity of the solution [22] and is less sensitive to different levels of noise [21]. In this paper, we extend ELM regression algorithm based on this variant.

### 3. Missing Data Problem in ELM Learning

*3.1. Missing Data Problem.* Nowadays, with ever-increasing data velocity and volume, missing data becomes a common phenomenon. Generally, there are two missing patterns, that is, missing feature and missing label. In this paper, we focus on the issue of missing feature.

From the causes of missing data, there are two circumstances. In the first circumstance, the missing features exist but their values are unobserved for the reason that information is lost or some features are too costly to be acquired [23]. Examples of such case can be found in many domains. Sensors in a remote sensor network may be damaged and fail to collect data intermittently. Certain regions of a gene micro array may fail to yield measurements of the underlying gene expressions due to scratches, fingerprints, or dust [16]. Second is inherently missing. In this circumstance, different samples inherently contain different features. For instance, in packed malware identification, instances contain some unreasonable values. In the web-page task, one useful feature of a given page may be the most common topic of other sites that point to it. If this particular page has no such parents, however, the feature is null, and should be considered structurally missing [24]. Obviously, imputation for this circumstance is meaningless.

*3.2. Traditional Approaches for Missing Data Learning.* Generally, there are three approaches for dealing with missing features in machine learning. The first approach is omitting, which includes sample deletion and feature filtering. Sample deletion simply omits the samples containing missing features and applies standard learning algorithms in the remaining samples. An example is shown in Figure 2; *sample 2* with two missing features is deleted. Feature filtering omits the features that are missing in most samples. Figure 3 interprets this approach. Obviously, the advantage of omitting based approaches is simple and computationally inexpensive. Notably, the key point of omitting is keeping as much as possible useful information while omitting. But it is difficult to do that. Both of them inevitably omit some useful information. When there is massive information retained after being partly omitted, this approach can be a better choice. Otherwise, in the situation of much useful information being omitted while few being retained, this kind of approaches affects learning precision seriously. Second approach is imputation. In data preprocessing phase, missing features are filled with most possible values [25]. Simple imputations fill the missing features with some default value such as zero or average value of other samples. Complex imputations use some probabilistic density function or distribution function to estimate the missing features. The computational complexity of imputation varies with different estimation methods. Imputation makes sense when the features are known to

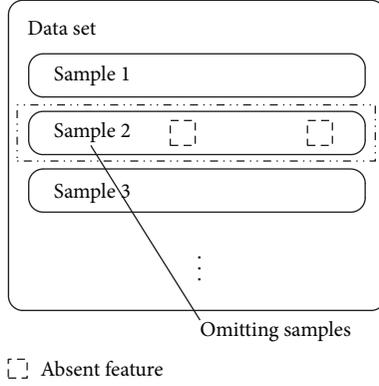


FIGURE 2: Sample deletion.

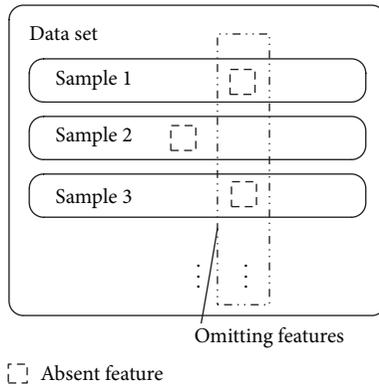


FIGURE 3: Feature filtering.

exist and relatively easy to be estimated. But in some cases, the missing values are difficult to impute. Besides, in the situation of inherent missing, imputation is meaningless. Last but promising approach appears in recent years. Researchers manage to extend standard learning algorithms to deal with missing features. Different from the above two, it needs no extra preprocessing for missing data. It keeps the intactness of incomplete samples and extends standard learning algorithms according to their learning theory. Consequently, the extended algorithms learn over complete samples as well as incomplete samples. In this paper, the proposed sample-based learning belongs to this category.

#### 4. Sample-Based ELM Framework and Algorithms

In this section, we start by illustrating ELM learning in a max-margin way as Huang explains and formalizes the learning problem with nonexistent features. Then, we explain why standard max-margin learning must be modified for learning missing data. Based on this, we propose the S-ELM classification algorithm in Section 4.2 and the S-ELM  $\epsilon$ -insensitive regression algorithm in Section 4.3.

**4.1. Sample-Based Learning Framework.** In the viewpoint of geometric formulation, maximizing margin is equivalent to maximizing the dot products of sample vectors and output

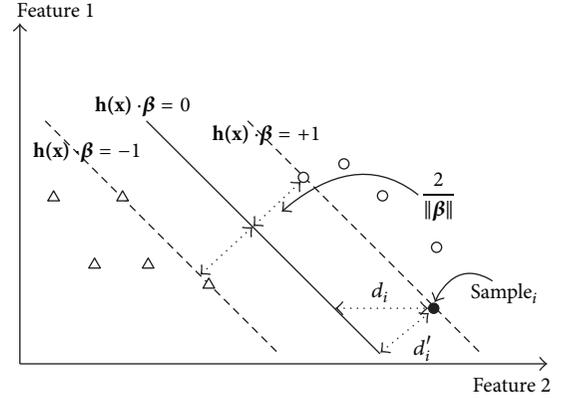


FIGURE 4: ELM's max margin in feature space.

weight vectors. Meanwhile, the smaller the norms of output weights are, the better generalization the network tends to have. ELM learning is the process of adjusting output weight vectors by the above two objectives. For samples containing missing features, the margins are incorrectly scaled [24]. For example, in Figure 4, *sample i* misses feature 1. The distance between *sample i* and separating plane should be calculated only in existing features (i.e.,  $d_i$ ). However, in standard ELM, distance is calculated in full space. So  $d_i$  is incorrectly scaled up, that is,  $d_i'$ . Based on this observation, we are inspired to consider each sample in its own relevant subspace. This is just the essence of sample-based learning framework. Samples with missing features should be treated as residing in the relevant subspaces rather than in full space. The optimization formula of S-ELM is naturally derived as follows:

$$\min_{\beta} \|\beta^{(i)}\|_p^{\sigma_1} + C \times \|\mathbf{H}\beta^{(i)} - \mathbf{Y}\|_q^{\sigma_2}. \quad (11)$$

Compared with standard general ELM, the essential transformation is substitution original output weight  $\beta$  with sample-based output weight  $\beta^{(i)}$ . Specifically speaking, there are two objectives in minimization optimization formula (i.e., empirical risk and structural risk). Missing features affect the calculation of empirical risk (i.e., dot product of output weight and sample vector). Fortunately, considering distances in sample-based subspaces, we keep the two optimization objectives consistent by transforming  $\beta$  into  $\beta^{(i)}$ . Actually,  $\beta^{(i)}$  is the  $\beta$ 's projection in *sample i*'s relevant subspace. For samples without missing feature, sample-based output weights are just the same with standard one.

**4.2. S-ELM Classification Algorithm.** According to sample-based learning framework, the optimization formula of sample-based ELM(S-ELM) classification is constructed as (12). Here, the extension is based on (4):

$$\begin{aligned} \min_{\beta^{(i)}, \xi_i} & \frac{1}{2} \|\beta^{(i)}\|^2 + C \sum_{i=1}^n \xi_i, \\ \text{s.t.} & \mathbf{y}_i * (\beta^{(i)T} \cdot \mathbf{x}_i) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \quad (12)$$

As it can be seen, different missingness among samples may induce different  $\beta^{(i)}$ s. In order to guarantee better generalization, the maximum of all  $\beta^{(i)}$ s should be minimized. Consequently, (12) can be transformed into

$$\begin{aligned} \min_{\beta^{(i)}, \xi_i} & \left( \max_{1 \leq i \leq n} \frac{1}{2} \|\beta^{(i)}\|^2 \right) + C \sum_{i=1}^n \xi_i, \\ \text{s.t.} & \mathbf{y}_i * \left( \beta^{(i)T} \cdot \mathbf{x}_i \right) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \quad (13)$$

By introducing an indication matrix  $\mathbf{S}$  (each of its components  $s_{i,j} \in \{0, 1\}^{n \times d}$  indicates whether the corresponding feature of sample is missing or not), (13) can be equivalently rewritten as

$$\begin{aligned} \min_{\beta, \xi_i} & \max_{1 \leq i \leq n} \frac{1}{2} \sum_{p=1}^d \mathbf{S}(i, p) \beta_p^2 + C \sum_{i=1}^n \xi_i, \\ \text{s.t.} & \mathbf{y}_i * \left( \sum_{p=1}^d \mathbf{S}(i, p) \beta_p \cdot \mathbf{x}_{i,p} \right) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \quad (14)$$

Since (14) is not a convex optimization problem, it is more difficult to be solved than standard ELM. Fortunately, we can transform it into a convex one (i.e., (15)) by using an auxiliary variable  $\theta$  ( $\theta = \max_{1 \leq i \leq n} (1/2) \sum_{p=1}^d \mathbf{S}(i, p) \beta_p^2$ ). Consider the following:

$$\begin{aligned} \min_{\beta, \xi_i, \theta} & \theta + C \sum_{i=1}^n \xi_i \\ \text{s.t.} & \mathbf{y}_i * \left( \sum_{p=1}^d \mathbf{S}(i, p) \beta_p \cdot \mathbf{x}_{i,p} \right) \geq 1 - \xi_i, \\ & \theta \geq \frac{1}{2} \sum_{p=1}^d \mathbf{S}(i, p) \beta_p^2, \\ & \xi_i \geq 0. \end{aligned} \quad (15)$$

From (15), it can be found that (1) the objective is a quadratic function, which is convex; (2) the first constraint is linear in variables, which is also convex; and (3) the second constraint is a quadratic cone, which is again convex. Therefore, (15) is a convex optimization problem, which can be efficiently solved by many off-the-shelf convex optimization packages. In this paper, we use monqp package to solve it. Further, (15) is a quadratic constraint quadratic programming (QCQP) problem; more advanced optimization techniques such as Nesterov's methods [26] can be applied to improve its computation efficiency.

**4.3. S-ELM  $\epsilon$ -Insensitive Regression Algorithm.** In this section, we extend the  $\epsilon$ -insensitive ELM [21] based on the sample-based learning framework. Based on (9), we substitute

standard output weight with the sample-based one. The resultant optimization formula is

$$\begin{aligned} \min_{\beta^{(i)}, \xi_i, \xi_i^*} & \frac{1}{2} \|\beta^{(i)}\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*), \\ \text{s.t.} & \mathbf{y}_i - \beta^{(i)T} \mathbf{x}_i \leq \epsilon + \xi_i, \\ & \beta^{(i)T} \mathbf{x}_i - \mathbf{y}_i \leq \epsilon + \xi_i^*. \end{aligned} \quad (16)$$

In (16),  $\beta^{(i)}$  corresponding to different samples may have different components. So we minimize their maximum value. We change its optimization objective as (17). Then, we utilize the same transformation technique in Section 4.2 and derive the final convex optimization. In specific, we list the transformation process as follows:

$$\begin{aligned} \min_{\beta^{(i)}, \xi_i, \xi_i^*} & \left( \max_i \frac{1}{2} \|\beta^{(i)}\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \right), \\ \text{s.t.} & \mathbf{y}_i - \beta^{(i)T} \mathbf{x}_i \leq \epsilon + \xi_i, \\ & \beta^{(i)T} \mathbf{x}_i - \mathbf{y}_i \leq \epsilon + \xi_i^*, \end{aligned} \quad (17)$$

$$\begin{aligned} \min_{\beta^{(i)}, \xi_i, \xi_i^*} & \theta + C \sum_{i=1}^n (\xi_i + \xi_i^*), \\ \text{s.t.} & \mathbf{y}_i - \beta^{(i)T} \mathbf{x}_i \leq \epsilon + \xi_i, \\ & \beta^{(i)T} \mathbf{x}_i - \mathbf{y}_i \leq \epsilon + \xi_i^*, \\ & \theta \geq \frac{1}{2} \sum_{p=1}^d \mathbf{S}(i, p) \beta_p^2. \end{aligned} \quad (18)$$

From (18), we can see that (1) the objective is convex; (2) the first and third constraint is linear inequality which is convex; (3) the second constraint is a quadratic inequality which is again convex. Consequently, (18) is a convex optimization problem. It can be efficiently solved by many existing methods.

## 5. Experiments

In this section, We demonstrate the proposed algorithms over synthetic data sets, UCI benchmark data sets, and a real world fingerprint image data set to validate the effectiveness of S-ELM learning framework. Missing data used in experiments is produced artificially. In detail, we generate indication matrix (which is depicted in Section 4.2) for each raw data set by the same numbers of row and column. First, all the components of indication matrix are initialized with one. Then, according to specific missing ratio, some randomly chosen components are set to zero. Note that we make sure there is no single row or column to be set all to zeros. For the same data set, indication matrixes are produced for training and testing data, respectively.

To validate the effectiveness of the proposed algorithms, we compare the performance of proposed algorithms with

TABLE 1: The UCI data sets used in classification.

| Dataset       | Number of samples | Number of features |
|---------------|-------------------|--------------------|
| Breast cancer | 286               | 9                  |
| Diabetes      | 768               | 20                 |
| Flare solar   | 1389              | 10                 |
| German        | 1000              | 20                 |
| Splice        | 3190              | 61                 |
| Waveform      | 5000              | 21                 |

two imputation approaches (i.e., zero-filling and mean-filling). In specific, S-ELM learning algorithms run on the data set with missing data while standard ELM algorithms run in the imputed data set. For abbreviation, ZF-ELM represents standard ELM algorithm running in zero-filling data set, and MF-ELM represents standard ELM algorithm running in mean-filling data set.

*5.1. Experiment Settings.* All our experiments are implemented on MATLAB 2013b environment running in Core 3.0 GHz CPU and 8 GB RAM. Prior to each algorithm execution, we do some data preprocessing. First, we normalize data sets to  $[0, 1]$ . Then, we permute data sets randomly and divide them into training data and testing data (#train/#test equals to 3/2). In order to eliminate randomness, final results are the average of fifty times repetitions. We also calculate the standard deviation to show the stability of algorithms.

*5.2. Results of S-ELM Classification.* In this section, we use aggregated classification accuracy as the standard to measure the learning precision of different algorithms. Missing ratios are between 0.1 and 0.6 with 0.1 interval. For each algorithm and each data set, the regularization factor  $C$  is chosen by fivefold cross validations. The UCI benchmark data sets are varied in the number of features and samples; detailed information is shown in Table 1.

The classification accuracy of three algorithms is plotted in Figure 5. As it can be observed, the curves corresponding to S-ELM are above other two in most cases, indicating its superior performance. Furthermore, Table 2 lists the average accuracy and standard deviation. In general, for various UCI benchmark data sets, the S-ELM classification algorithm achieves better accuracy and stability over different missing ratios. The results are consistent with our analysis in Section 4. The two imputation methods classify all samples in full feature space, while S-ELM learning assumes samples with the different vector components lying in their own relevant feature spaces. Particularly, compared with original ELM, S-ELM classification algorithm does not bring extra computational cost.

*5.3. Fingerprint Classification.* As an application, we evaluate our methods in a real world problem, that is, automatic fingerprint classification, which is an effective index scheme for reducing matching candidate in fingerprint identification. Through gross classification, a query finger is only compared with samples of the same class rather than all samples in the

TABLE 2: Classification performance comparisons over different missing ratios on the UCI benchmark data sets. Performance includes mean aggregate accuracy  $\pm$  standard deviation and training time (in seconds).

|               | S-ELM                         | ZF-ELM                        | MF-ELM                        |
|---------------|-------------------------------|-------------------------------|-------------------------------|
| Breast cancer | $0.7185 \pm 0.0151$<br>0.3473 | $0.6570 \pm 0.0334$<br>0.2776 | $0.6901 \pm 0.0183$<br>0.2947 |
| Diabetes      | $0.7375 \pm 0.0263$<br>0.9315 | $0.6852 \pm 0.0467$<br>0.7257 | $0.6957 \pm 0.0448$<br>0.7112 |
| Flare solar   | $0.5635 \pm 0.0217$<br>2.938  | $0.5369 \pm 0.0348$<br>2.108  | $0.5454 \pm 0.0441$<br>2.272  |
| German        | $0.7624 \pm 0.0250$<br>3.748  | $0.6984 \pm 0.0301$<br>2.327  | $0.7150 \pm 0.0257$<br>2.374  |
| Splice        | $0.6704 \pm 0.0602$<br>10.83  | $0.5734 \pm 0.0275$<br>11.26  | $0.5916 \pm 0.0273$<br>9.273  |
| Waveform      | $0.8287 \pm 0.0236$<br>120.5  | $0.7206 \pm 0.0271$<br>99.26  | $0.7456 \pm 0.0314$<br>108.5  |

whole database. It reduce computational complexity greatly. In practice, there are some meaningless features in fingerprint data set due to distortion, wet and dry impression, and so forth. The meaningless features can be seen as a kind of missing data. Therefore, we apply the S-ELM classification algorithm to this application.

The fingerprint image data set used in our experiment contains 2000 samples. First, we categorize all those fingerprints into three basic patterns manually: loop, whorl, and arch. All fingerprint images are 8-bit gray-level bmp files and the image resolution is  $328 * 356$ . Then, we use the first 1000 pairs of fingerprints for training and the second 1000 pairs of fingerprints for testing. Through fingerprint image preprocessing (e.g., edge detection, segmentation, feature, thinning, and extraction), each fingerprint image is expressed as a feature vector of 400 dimensions.

Too low accuracy does not make sense in the application of fingerprint classification. We set missing ratios in a relatively low range, that is, 0.1, 0.2, and 0.3. Besides, zero filling is meaningless. We only take MF-ELM as comparison. The fingerprint data set contains three classes of samples. We use one-versus-rest method to classify them, respectively; that is, we take a target from three classes each time. All the classification accuracy list in Table 3 is the average values of ten times execution results. It can be observed that (1) the classification accuracy of fingerprint classification seriously affected by missing data; (2) the computational cost of different missing ratios are basically the same; (3) in almost all conditions, S-ELM classification algorithm performs more accurately and stably than MF-ELM with almost the same computational cost.

*5.4. Results of S-ELM  $\epsilon$ -Insensitive Regression.* In this section, we evaluate the S-ELM  $\epsilon$ -insensitive regression algorithm in synthetic data sets and UCI benchmark data sets. Root mean

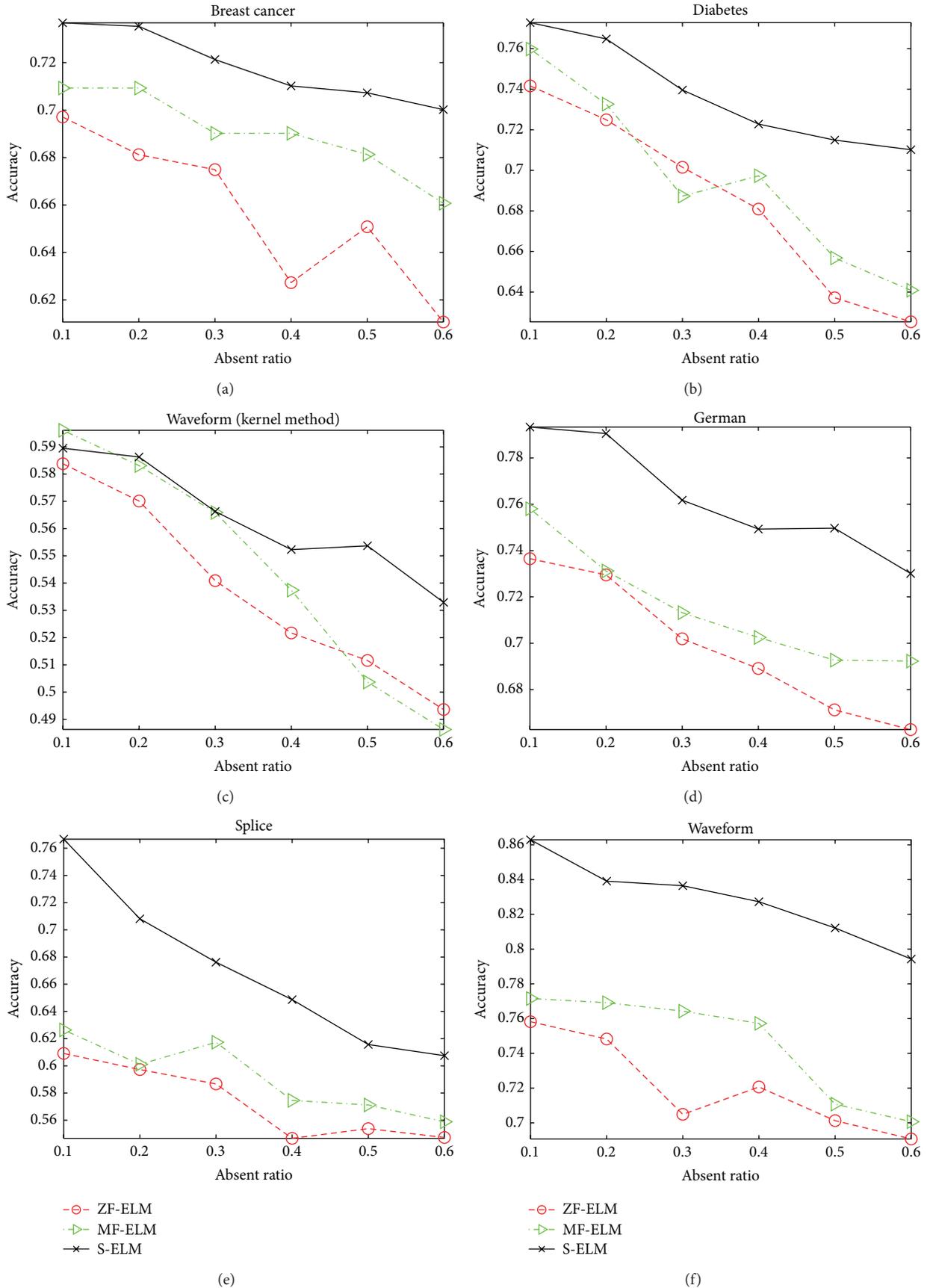


FIGURE 5: Classification accuracy comparison in the UCI data sets.

TABLE 3: Performance comparison of S-ELM and MF-ELM over different missing ratios on the fingerprint data set.

| Target class | Missing ratio | S-ELM               |             | MF-ELM              |             |
|--------------|---------------|---------------------|-------------|---------------------|-------------|
|              |               | Accuracy $\pm$ std. | Time (Sec.) | Accuracy $\pm$ std. | Time (Sec.) |
| Loop         | 10%           | 0.8256 $\pm$ 0.0736 | 89.37       | 0.7967 $\pm$ 0.0985 | 85.72       |
|              | 20%           | 0.8029 $\pm$ 0.1253 | 85.91       | 0.7265 $\pm$ 0.1371 | 86.28       |
|              | 30%           | 0.7694 $\pm$ 0.1191 | 90.43       | 0.6303 $\pm$ 0.1039 | 87.93       |
| Whorl        | 10%           | 0.8137 $\pm$ 0.0627 | 67.28       | 0.8129 $\pm$ 0.0922 | 60.72       |
|              | 20%           | 0.7436 $\pm$ 0.0963 | 71.32       | 0.7269 $\pm$ 0.1167 | 67.49       |
|              | 30%           | 0.7102 $\pm$ 0.0849 | 69.09       | 0.6359 $\pm$ 0.1076 | 68.25       |
| Arch         | 10%           | 0.8361 $\pm$ 0.0852 | 102.7       | 0.8057 $\pm$ 0.0937 | 94.73       |
|              | 20%           | 0.7659 $\pm$ 0.0974 | 98.31       | 0.7561 $\pm$ 0.1170 | 96.83       |
|              | 30%           | 0.7192 $\pm$ 0.0827 | 100.3       | 0.6574 $\pm$ 0.1207 | 96.67       |

TABLE 4: Synthetic data sets and corresponding functions.

| Data set             | Function                                                                                                                      |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Synthetic data set 1 | $f_1(x) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$                                                                                  |
| Synthetic data set 2 | $f_2(x) = x_1 \times x_2 \times x_3 + x_2 \times x_3 \times x_4$<br>$+ x_3 \times x_4 \times x_5 + x_4 \times x_5 \times x_6$ |
| Synthetic data set 3 | $f_3(x) = 2^{x_1} + 2^{x_2} + 2^{x_3} + 2^{x_4} + 2^{x_5} + 2^{x_6}$                                                          |
| Synthetic data set 4 | $f_4(x) = \frac{x_1 + x_2 + x_3}{x_4 + x_5 + x_6}$                                                                            |

TABLE 5: The UCI benchmark data sets for regression.

| Dataset            | # train | # test | # features |
|--------------------|---------|--------|------------|
| Body fat           | 152     | 100    | 14         |
| Housing            | 304     | 202    | 13         |
| Pyrim              | 45      | 29     | 27         |
| Abalone            | 2507    | 1670   | 8          |
| Bike sharing       | 439     | 293    | 6          |
| Airfoil self-noise | 901     | 602    | 5          |

square error (RMSE) is used as the measurement to evaluate the prediction accuracy of algorithms. It is calculated as

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^n (\hat{y}_t - y_t)^2}{n}}. \quad (19)$$

We generate synthetic data sets by combinations of basic algebraic operation functions. For each function, all independent variables' domains are in  $[0, 1]$ , and their intervals are randomly different. The detail is given in Table 4. As for S-ELM  $\epsilon$ -insensitive regression,  $\epsilon$  is set to be 0.01 empirically.

From Figure 6, it can be observed that the S-ELM  $\epsilon$ -insensitive regression algorithm achieves better prediction accuracy in different missing ratios. Specifically, S-ELM performs better than ZF-ELM, which proves that S-ELM reduces generalization error by using sample-based output weight vector norm. S-ELM beats MF-ELM with comparatively less obvious advantage. It shows that mean value is closer to missing value than zero.

To further illustrate the advantage of the S-ELM  $\epsilon$ -insensitive regression algorithm, we use UCI benchmark data sets for comparison. Table 5 specifies the UCI benchmark data sets used in our experiments. Those four data sets are varied in scales.

Mean RMSE and standard deviation are reported in Table 6. S-ELM regression algorithm performs better than ZF-ELM and MF-ELM over different missing ratios. As Figure 7 shows, with the missing ratio increasing, both ZF-ELM and MF-ELM's prediction accuracies decrease rapidly,

TABLE 6: Performance comparisons with UCI benchmark data sets. The two rows of each cell represent mean RMSE  $\pm$  standard deviation over different missing ratios and training time (in seconds).

|                    | S-ELM               | ZF-ELM              | MF-ELM              |
|--------------------|---------------------|---------------------|---------------------|
| Body fat           | 0.1299 $\pm$ 0.0905 | 0.1781 $\pm$ 0.1007 | 0.1457 $\pm$ 0.1088 |
|                    | 0.1560              | 0.1758              | 0.1872              |
| Housing            | 0.2229 $\pm$ 0.1019 | 0.3234 $\pm$ 0.1697 | 0.2511 $\pm$ 0.1261 |
|                    | 0.5628              | 0.3432              | 0.3869              |
| Pyrim              | 0.3816 $\pm$ 0.0842 | 0.5603 $\pm$ 0.2253 | 0.4537 $\pm$ 0.1396 |
|                    | 0.0624              | 0.1008              | 0.0926              |
| Abalone            | 0.2336 $\pm$ 0.1083 | 0.3004 $\pm$ 0.1537 | 0.2643 $\pm$ 0.1193 |
|                    | 213.5               | 177.6               | 182.4               |
| Bike sharing       | 0.3886 $\pm$ 0.1020 | 0.4662 $\pm$ 0.1596 | 0.3977 $\pm$ 0.1218 |
|                    | 7.263               | 7.109               | 6.993               |
| Airfoil self-noise | 0.3765 $\pm$ 0.1591 | 0.4563 $\pm$ 0.2470 | 0.3810 $\pm$ 0.1928 |
|                    | 52.38               | 50.74               | 51.61               |

while S-ELM decreases mildly. The most obvious advantage of S-ELM appears in Pyrim data set, especially when missing ratio is high. It proves that it is more difficult for imputation approaches to estimate the true value of missing features in small scale data set.

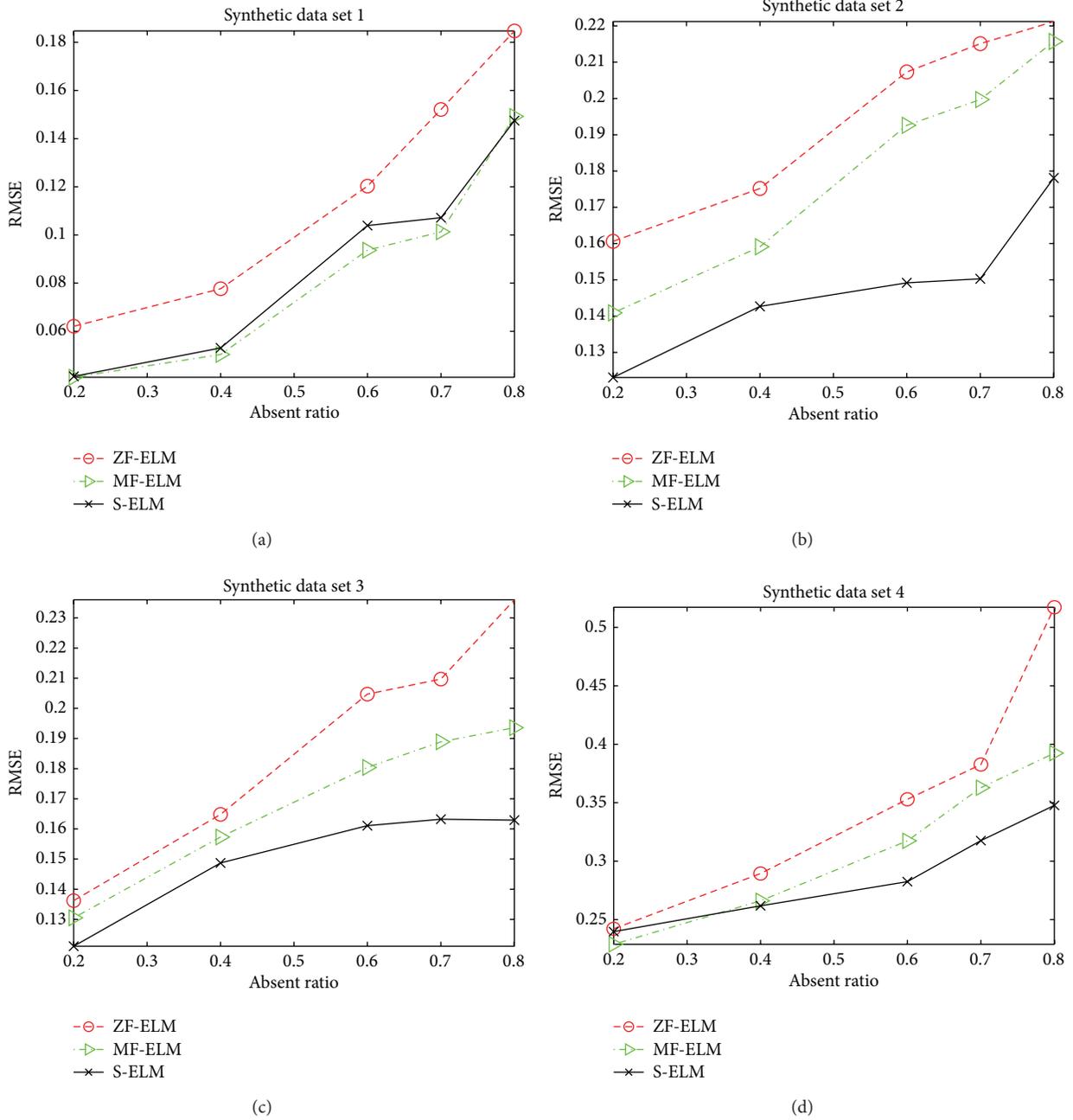


FIGURE 6: RMSE comparisons in the synthetic data sets.

## 6. Conclusions and Future Work

In this paper, we propose a sample-based learning framework for ELM in order to cope with the problem of missing data. Further, we extend ELM classification and  $\epsilon$ -insensitive ELM regression in this framework. For the proposed algorithms, we transform their formulations into convex optimizations. We compare proposed algorithms with two widely used imputation methods in synthetic, UCI benchmark data sets and a real world fingerprint image data set. Result shows

that S-ELM achieves better classification and regression accuracy and stability without introducing extra computational complexity. Particularly, S-ELM shows remarkable advantage when the missing ratio is relatively high. Many following works are worth exploring. In the next, we are going to extend more ELM algorithms based on sample-based learning framework. Meanwhile, we will handle missing data in feature selection to improve learning accuracy [27]. Further, we will consider missing data in multiple data sources learning.

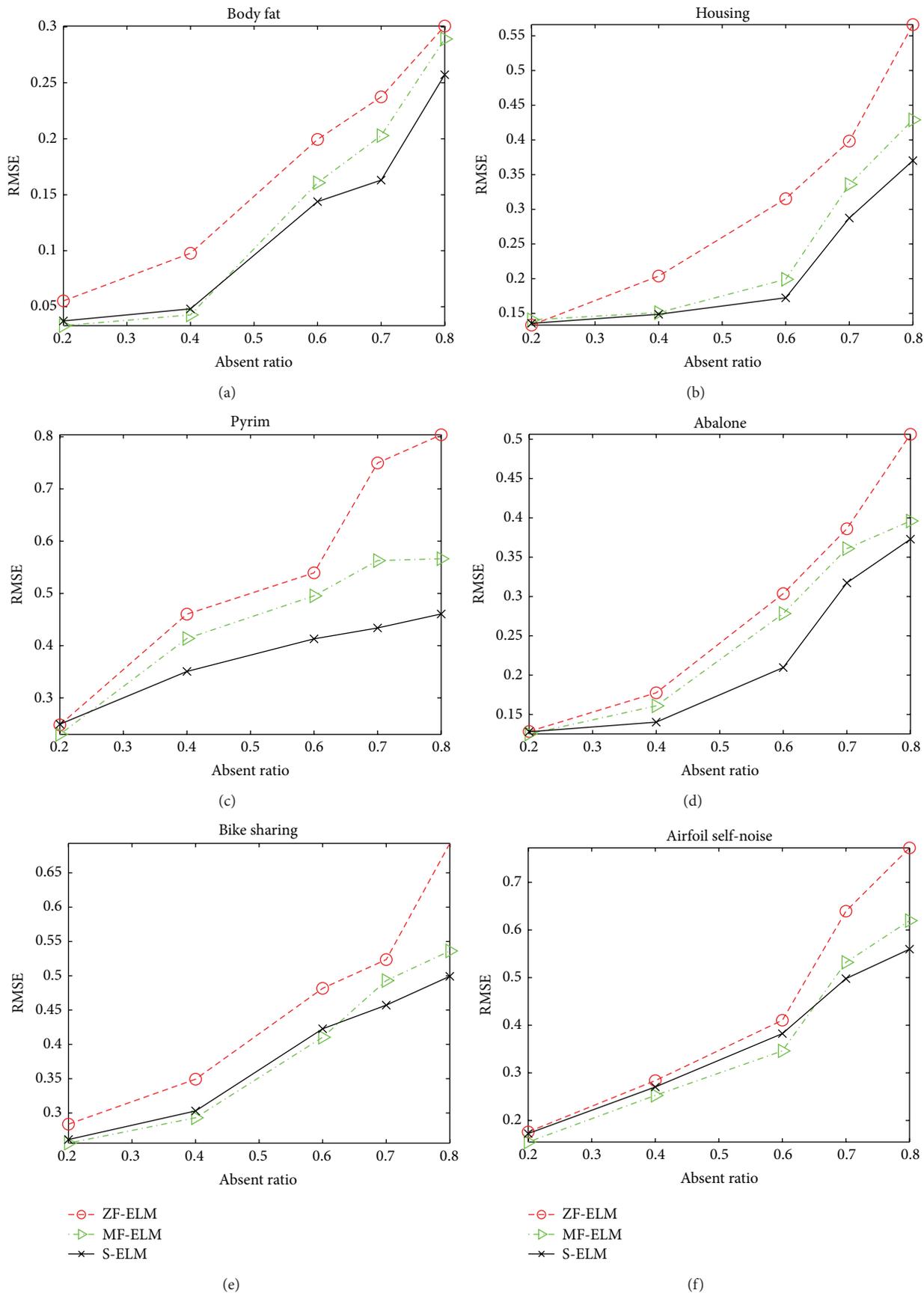


FIGURE 7: RMSE comparisons in the UCI benchmark data sets.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is supported by the Major State Basic Research Development Program of China (973 Program) under the Grant no. 2014CB340303, the National High Technology Research and Development Program of China (863 Program) under Grant no. 2013AA01A213, and the National Natural Science Foundation of China under Grant no. 61402490.

## References

- [1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, Budapest, Hungary, July 2004.
- [2] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [3] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [4] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09)*, pp. 389–395, Nashville, Tenn, USA, April 2009.
- [5] J. M. Martínez-Martínez, P. Escandell-Montero, E. Soria-Olivas, J. D. Martín-Guerrero, R. Magdalena-Benedito, and J. Gómez-Sanchis, "Regularized extreme learning machine for regression problems," *Neurocomputing*, vol. 74, no. 17, pp. 3716–3721, 2011.
- [6] J. Luo, C.-M. Vong, and P.-K. Wong, "Sparse bayesian extreme learning machine for multi-classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 836–843, 2014.
- [7] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization," *Neurocomputing*, vol. 74, no. 16, pp. 2413–2421, 2011.
- [8] A. Mozaffari and N. L. Azad, "Optimally pruned extreme learning machine with ensemble of regularization techniques and negative correlation penalty applied to automotive engine coldstart hydrocarbon emission identification," *Neurocomputing*, vol. 131, pp. 143–156, 2014.
- [9] Y. Lan, Y. C. Soh, and G.-B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16–18, pp. 3028–3038, 2010.
- [10] X. Liu, L. Wang, G.-B. Huang, J. Zhang, and J. Yin, "Multiple kernel extreme learning machine," *Neurocomputing*, vol. 149, part A, pp. 253–264, 2015.
- [11] N.-Y. Liang, G.-B. Huang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "A fast and accurate on-line sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.
- [12] Y. Lan, Y. C. Soh, and G.-B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, no. 13–15, pp. 3391–3395, 2009.
- [13] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.
- [14] Z. Deng, K.-S. Choi, L. Cao, and S. Wang, "T2fela: type-2 fuzzy extreme learning algorithm for fast training of interval type-2 TSK fuzzy logic system," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 664–676, 2014.
- [15] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on mapreduce," *Neurocomputing*, vol. 102, pp. 52–58, 2013.
- [16] B. M. Marlin, *Missing data problems in machine learning [Ph.D. thesis]*, University of Toronto, 2008.
- [17] P. Royston, "Multiple imputation of missing values," *Stata Journal*, vol. 4, pp. 227–241, 2004.
- [18] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16–18, pp. 3460–3468, 2008.
- [19] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [20] X. Liu, L. Wang, J. Yin, and L. Liu, "Incorporation of radius-info can be simple with SimpleMKL," *Neurocomputing*, vol. 89, pp. 30–38, 2012.
- [21] S. Balasundaram, "On extreme learning machine for  $\epsilon$ -insensitive regression in the primal by Newton method," *Neural Computing and Applications*, vol. 22, no. 3-4, pp. 559–567, 2013.
- [22] J. Goeman, R. Meijer, and N. Chaturvedi, "L1 and L2 penalized regression models," cran.r-project.or, 2012.
- [23] J. A. R. Little, "Regression with missing x's: a review," *Journal of the American Statistical Association*, vol. 87, no. 420, pp. 1227–1237, 1992.
- [24] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, "Max-margin classification of data with absent features," *The Journal of Machine Learning Research*, vol. 9, pp. 1–21, 2008.
- [25] N. J. Horton and S. R. Lipsitz, "Multiple imputation in practice: comparison of software packages for regression models with missing variables," *The American Statistician*, vol. 55, no. 3, pp. 244–254, 2001.
- [26] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87 of *Applied Optimization*, Springer, Amsterdam, The Netherlands, 2004.
- [27] X. Liu, L. Wang, J. Zhang, J. Yin, and H. Liu, "Global and local structure preservation for feature selection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1083–1095, 2014.

## Research Article

# Optimization ELM Based on Rough Set for Predicting the Label of Military Simulation Data

**Xiao-jian Ding and Ming Lei**

*Science and Technology on Information Systems Engineering Laboratory, Nanjing 210007, China*

Correspondence should be addressed to Xiao-jian Ding; [wjswsl@163.com](mailto:wjswsl@163.com)

Received 19 April 2014; Revised 26 July 2014; Accepted 18 August 2014; Published 25 September 2014

Academic Editor: Yi Jin

Copyright © 2014 X.-j. Ding and M. Lei. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

By combining rough set theory with optimization extreme learning machine (OELM), a new hybrid machine learning technique is introduced for military simulation data classification in this study. First, multivariate discretization method is implemented to convert continuous military simulation data into discrete data. Then, rough set theory is employed to generate the simple rules and to remove irrelevant and redundant variables. Finally, OELM is compared with classical extreme learning machine (ELM) and support vector machine (SVM) to evaluate the performance of both original and reduced military simulation datasets. Experimental results demonstrate that, with the help of RS strategy, OELM can significantly improve the testing rate of military simulation data. Additionally, OELM is less sensitive to model parameters and can be modeled easily.

## 1. Introduction

Information warfare, which is the competitive use of information in modern forces, has recently become of increasing importance to the military domain. Information superiority and knowledge dominance are key points to winning the future war. There has been a substantial growth in the amount of stored information in military systems, mainly due to the rapid development of military information construction. However, the large amount of stored data becomes impracticable for specialists to analyze through conventional methods.

Military data research has gained great importance and interest in recent years, especially in operational areas. Data learning technology provides important analysis and decision services to aid the commander. To provide decision support for commanders, it is worth studying how to effectively learn the massive data that emerges in operations. The data used in the equipment war gaming system was classified on the basis of introducing terms about verification, validation, and certification (VV&C) and a sort criterion of authoritative data sources (ADS) was also put forward, as described by

Liguo et al. [1]. Li et al. [2] analyzed the characteristics of military data, put forward methods to classify data based on their characteristics and functions, and discussed data sources for combat simulation. Gao et al. [3] introduced a military simulation data aggregation framework for extraction, transformation, transportation, and loading based on message oriented middleware. As seen in [4], preliminary study on simulation data mining by rough set (RS) theory was done. RS enables us to find the dependencies and to reduce the number of attributes contained in military simulation data. With the help of RS, valuable results can be obtained by mining the battle simulation data, and the attributes which make a strong impact on performance of datasets learning can be found. However, given newly generated battle data, one cannot predict the classification results with the method of [4].

In this study, a new regularization algorithm-optimization extreme learning machine (OELM) [5], together with RS theory, is selected as main components for constructing an integrated predictor. OELM is an optimization algorithm based on conventional extreme learning machine (ELM) [6–11], which has the following

properties: first, compared to the traditional ELM, the minimization norm of output weights enables OELM to get better generalization performance; second, compared to support vector machine (SVM) [12], OELM finds the optimal solution in the search space of  $[0, C]^N$ , where SVM always searches the suboptimal solution of OELM due to equation constraints, so OELM usually achieves better generalization performance than a traditional SVM; third, OELM is less sensitive to specified parameters and can be implemented easily.

In this paper, the performance of an integrated method of multivariate discretization, RS theory, and OELM for military simulation data analysis is investigated and compared with state-of-the-art classifiers: ELM and SVM. As there exist some interactions between different attributes of military simulation data, multivariate discretization attempts to find the correct cuts by taking into account one attribute independently from the others. RS theory is adopted to reduce redundant attributes of military simulation data, and then OELM algorithm is employed to train this new dataset with optimal subset attributes.

We start this work with description of the integrated method in Section 2. Military simulation data preprocessing method is discussed in Section 3. Military simulation data analysis is shown in Section 4. Section 5 presents the experimental setup and details the results arrived at. Finally, Section 6 gives the conclusion.

## 2. Integrated Method

In this section, RS theory is adopted to reduce redundant attributes of the military simulation dataset, and then OELM algorithm is employed to train this new dataset with optimal attributes subset. The main contributions of this work are as follows: (1) multivariate discretization method is used to preprocess the original military dataset; (2) rules generated by RS method are used to create a reduced military dataset; (3) OELM, together with two other popular algorithms, is used to measure the generalization performance of both original dataset and reduced dataset.

**2.1. OELM Review.** For  $N$  arbitrary distinct samples  $\{\mathbf{x}_i, t_i\}_{i=1}^N$ , where  $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{id}] \in \mathbf{R}^d$  and  $t_i \in \{-1, 1\}$ , the decision function given by an ELM is

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \right) = \text{sign}(\boldsymbol{\beta} \cdot h(\mathbf{x})), \quad (1)$$

where  $\beta_i$  is the output weight from the  $i$ th hidden node to the output node and  $G(\mathbf{a}_i, b_i, \mathbf{x})$  is the output of the  $i$ th hidden node.  $h(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]$  is the output vector of the hidden layer, and it can map the training data  $\mathbf{x}_i$  from the input space to the  $L$ -dimensional ELM feature space.

Based on the theory of Bartlett [13], feedforward neural network with smaller norm of weight, not only has smaller

training error, but also obtains better generalization performance. Therefore, ELM amounts to minimizing the training error with the minimized norm of the output weight:

$$\text{Minimize: } \sum_{i=1}^N \|\boldsymbol{\beta} \cdot h(\mathbf{x}_i) - t_i\| \quad (2)$$

$$\text{Minimize: } \|\boldsymbol{\beta}\|.$$

Any set of training data transformed from the input space to the ELM feature space with the mapping  $h(\mathbf{x})$  is linearly separable. In order to prevent the classification problem of overfitting, variables  $\xi_i, i = 1, \dots, N$  are introduced and one can minimize the testing error as follows:

$$\begin{aligned} \boldsymbol{\beta} \cdot h(\mathbf{x}_i) &\geq 1 - \xi_i \quad \text{for } t_i = +1, \\ \boldsymbol{\beta} \cdot h(\mathbf{x}_i) &\leq -1 + \xi_i \quad \text{for } t_i = -1. \end{aligned} \quad (3)$$

Two inequalities in (3) can be combined into one set of inequalities:

$$t_i \cdot \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i \quad \forall i. \quad (4)$$

Thus, OELM works based on solving the following quadratic program mathematical model:

$$\begin{aligned} \text{Minimize: } & \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^N \xi_i \\ \text{Subject to: } & t_i \cdot \boldsymbol{\beta} \cdot h(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \dots, N \\ & \xi_i \geq 0, \quad i = 1, \dots, N. \end{aligned} \quad (5)$$

Compared to SVM's optimization problem, there are three differences between SVM and OELM:

- (1) The mapping  $\phi(\mathbf{x}_i)$  used in SVM is usually unknown and cannot be computed directly. The function  $h(\mathbf{x}_i)$  used in OELM can be any bounded nonconstant piecewise continuous activation function for additive node, which can be calculated exactly.
- (2) Kernel parameters used in SVM need to be tuned manually, whereas all parameters of  $h(\mathbf{x})$  for OELM are selected randomly.
- (3) The bias  $b$  is not needed in the constraints of OELM's optimization problem, because the separating hyperplane of OELM tends to pass through the origin.

**2.2. Karush-Kuhn-Tucker Conditions.** According to Lagrange optimization theory, minimizing the norm of output weights is equivalent to solving the dual optimization problem of (5):

$$\text{Minimize: } \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N t_i t_j K_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) \alpha_i \alpha_j - \sum_{i=1}^N \alpha_i \quad (6)$$

$$\text{Subject to: } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, N.$$

ELM kernel matrix can be defined as

$$K_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j). \quad (7)$$

For the sake of convenience, (6) can be written compactly as

$$\begin{aligned} \text{Minimize: } & f(\boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\alpha}^T K \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\ \text{Subject to: } & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, N, \end{aligned} \quad (8)$$

where  $f : \mathbf{R}^N \rightarrow \mathbf{R}$  is a convex function,  $K = K_{\text{ELM}} \mathbf{t}^T \mathbf{t} \in \mathbf{R}^{N \times N}$  is positive semidefinite matrix,  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^T \in \mathbf{R}^N$ , and  $\mathbf{e} = [1, \dots, 1]^T \in \mathbf{R}^N$ .

Let  $g(\boldsymbol{\alpha}) = \partial f(\boldsymbol{\alpha}) / \partial \boldsymbol{\alpha} = K \boldsymbol{\alpha} - \mathbf{e}$ . Let  $g_i(\boldsymbol{\alpha})$  denote the  $i$ th element of  $g(\boldsymbol{\alpha})$ ; that is,  $g_i(\boldsymbol{\alpha}) = [K \boldsymbol{\alpha} - \mathbf{e}]_i$ . The Lagrange function of (8) is

$$L(\boldsymbol{\alpha}, \boldsymbol{\eta}, \boldsymbol{\rho}) = f(\boldsymbol{\alpha}) - \sum_{i=1}^N \eta_i \alpha_i - \sum_{i=1}^N \rho_i (C - \alpha_i), \quad (9)$$

where  $\eta_i$  and  $\rho_i$  are the Lagrange multipliers and are nonnegative values.

In order to find the optimal solutions of (9) we should have

$$\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\eta}, \boldsymbol{\rho})}{\partial \alpha_i} = 0 \implies g_i(\boldsymbol{\alpha}) - \eta_i + \rho_i = 0. \quad (10)$$

Based on the Karush-Kuhn-Tucker (KKT) theorem, the KKT conditions of (10) should be

primal feasibility

$$0 \leq \alpha_i \leq C, \quad \forall i; \quad (11)$$

dual feasibility

$$\eta_i \geq 0, \quad \forall i; \quad (12)$$

$$\rho_i \geq 0, \quad \forall i; \quad (13)$$

complementary slackness

$$\eta_i \alpha_i = 0, \quad \forall i; \quad (14)$$

$$\rho_i (C - \alpha_i) = 0, \quad \forall i. \quad (15)$$

Furthermore, consider the following.

- (1) If  $\alpha_i = 0$ , from (15), we have  $\rho_i = 0$ . Thus, from (10) and (12), we have

$$g_i(\boldsymbol{\alpha}) \geq 0. \quad (16)$$

- (2) If  $0 < \alpha_i < C$ , from (14) and (15), we have  $\eta_i = 0$  and  $\rho_i = 0$ . Thus, from (10), we have

$$g_i(\boldsymbol{\alpha}) = 0. \quad (17)$$

- (3) If  $\alpha_i = C$ , from (14), we have  $\eta_i = 0$ . Thus, from (10) and (13), we have

$$g_i(\boldsymbol{\alpha}) \leq 0. \quad (18)$$

The KKT conditions are both necessary and sufficient for optimality. Thus, (8) is solved when for all  $i$

$$\begin{aligned} \alpha_i = 0 & \iff g_i(\boldsymbol{\alpha}) \geq 0, \\ 0 < \alpha_i < C & \iff g_i(\boldsymbol{\alpha}) = 0, \\ \alpha_i = C & \iff g_i(\boldsymbol{\alpha}) \leq 0. \end{aligned} \quad (19)$$

Active set approach is the best choice to solve (8) because it is an iterative algorithm and it maintains feasibility of the vector  $\boldsymbol{\alpha}$  (i.e.,  $0 \leq \alpha_i \leq C$ ) while iteration continues until KKT conditions (19) are satisfied.

**2.3. Rough Set.** Rough set (RS) term, which was first introduced by Pawlak et al. [14] has become a popular pattern recognition tool for generating logical rules for prediction. Attribute reduction, which is one of the core parts of RS theory, means to find out the most informative attributes and remove the irrelevant or unimportant attributes with minimal information loss.

RS is the approximation of an uncertain set by a pair of precise concepts called lower and upper approximations. The lower approximation comprises elements belonging to it, whereas the upper approximation of the set includes elements which are possible members of the set.

RS model is defined in terms of an information system, which can be defined formally as 4-tuple in [15, 16]:

$$S = \langle U, Q, V, f \rangle, \quad (20)$$

where  $U = \{u_1, u_2, \dots, u_n\}$  is a nonempty set of objects called the universe,  $Q = \{q_1, q_2, \dots, q_n\}$  is a finite set of attributes,  $V_q$  is the domain of attribute  $q$ , and  $V = \bigcup_{q \in Q} V_q$ .  $f = U \times Q \rightarrow V$  is an information function that  $f(x, q) \in V_q$  for each  $q \in Q$ ,  $x \in U$ . Each object of universe is described by a vector

$$\text{Des}_q(x) = [f(x, q_1), f(x, q_2), \dots, f(x, q_m)], \quad (21)$$

where  $x \in U$ . To every nonempty subset of attributes  $P$  is associated with an indiscernibility relation on  $U$ , denoted by  $I_P$ :

$$I_P = \{(x, y) \in U \times U : f(x, q) = f(y, q), \forall q \in P\}. \quad (22)$$

Equation (22) is an equivalence relation. The family of all the equivalence classes of the  $I_P$  is denoted by  $U | I_P$  and class containing an element  $x$  by  $I_P(x)$ .

Formally, let  $X$  be a nonempty set of  $U$ . Set  $X$  is approximated by means of  $P$ -lower and  $P$ -upper approximations of  $X$ :

$$\begin{aligned} \underline{P}(X) &= \{x \in U : I_P(x) \subseteq X\}, \\ \overline{P}(X) &= \bigcup_{x \in X} I_P(x). \end{aligned} \quad (23)$$

The  $P$ -boundary of  $X$  is denoted by  $B_n(X)$

$$B_n(X) = \overline{P}(X) - \underline{P}(X). \quad (24)$$

The following relation holds:  $\underline{P}(X) \subseteq X \subseteq \overline{P}(X)$ .

There are three regions of interest, they are the inner region for objects in  $\underline{P}(X)$ , the outer region for objects not in  $\overline{P}(X)$ , and the boundary region for the objects that are uncertain.

TABLE 1: Parameters of simulation data.

|    | W1   | T1         | S    | V    | W2    | T2    | F      | P       | R    |
|----|------|------------|------|------|-------|-------|--------|---------|------|
| 1  | Fine | Flat       | Fast | More | Small | Short | Poor   | Normal  | High |
| 2  | Fine | Undulating | Slow | More | Small | Short | Poor   | Normal  | Low  |
| 3  | Fog  | Flat       | Slow | Less | Small | Long  | Better | Normal  | High |
| 4  | Fog  | Flat       | Fast | Less | Big   | Long  | Better | Nervous | Low  |
| 5  | Fine | Undulating | Slow | Less | Big   | Long  | Better | Normal  | High |
| 6  | Fine | Undulating | Fast | Less | Big   | Short | Poor   | Nervous | Low  |
| 7  | Fine | Flat       | Slow | More | Small | Short | Better | Nervous | High |
| 8  | Fog  | Flat       | Slow | More | Small | Short | Poor   | Nervous | Low  |
| 9  | Fine | Undulating | Fast | Less | Big   | Short | Better | Nervous | High |
| 10 | Fog  | Undulating | Fast | Less | Small | Long  | Better | Normal  | Low  |
| 11 | Fine | Flat       | Fast | Less | Small | Short | Better | Normal  | High |
| 12 | Fog  | Flat       | Fast | More | Big   | Short | Poor   | Normal  | Low  |

TABLE 2: Converted attribute of “Weather.”

| Attribute      | Fine | Fog |
|----------------|------|-----|
| Weather = fine | 1    | 0   |
| Weather = fog  | 0    | 1   |

### 3. Military Simulation Data Preprocessing

The performance of the integrated method is evaluated on countermeasure simulation data of tank formation [4], which was exercised on medium undulating ground, hills, and Taiwan highland for digital armored battalion simulation system. The specification of the dataset is presented in Table 1.

From Table 1, items “W1,” “T1,” . . . , and “R” represent “Weather,” “Terrain,” “Speed,” “Vegetation,” “Wind,” “Time,” “Formation,” “Psychology,” and “Shooting rate,” respectively. ELM algorithm can be applied only to datasets composed of categorical attributes but attributes in Table 1 are continuous variables. Discretization process is important for char variables because it is less prone to variance in estimation from small fragmented data and it provides better performance for rule extraction. The main discretization process methods are supervised [17], hierarchical [18], top-down [19], and multivariate [20]. The last one is employed in our work.

Multivariate discretization quantifies simultaneously many features. In the preprocessing phase, attribute “Weather” is converted into two new attributes “fine” and “fog,” which is shown in Table 2.

Based on examples shown in Table 2, other attributes are converted in Table 3.

### 4. Military Simulation Data Analysis

Generally speaking, it is a data mining task to distinguish the probability of shooting rate. The goal of military data analysis is to predict the unknown value of shooting rate, such as high or low. More specifically, military simulation data analysis process can be modeled as a classification problem.

A typical military dataset is sparse compared to a traditional classification dataset. It usually contains small samples but with many features. This kind of sparseness often

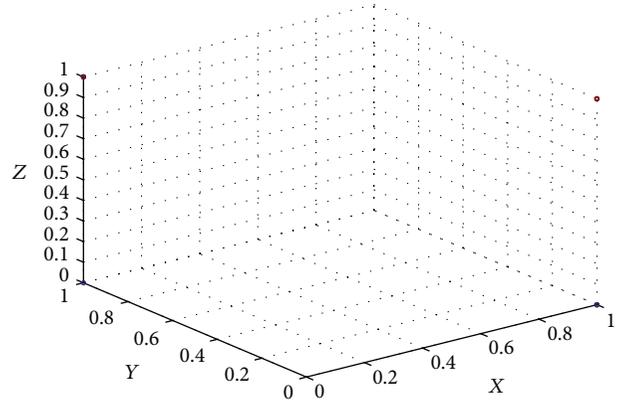


FIGURE 1: Military simulation data visualization.

descends the performance of some classifiers. To improve the classification rate, one can remove the irrelevant or redundant features. Intuitively speaking, a classifier trained in lower-dimensional feature space is expected to capture the inherent data distribution and performs better in such feature space.

There are many advantages of variable and feature selection, such as facilitating data visualization and data understanding, reducing training and testing time, and so forth. We give a 3-dimensional visualization of dataset discussed in Section 4, as seen in Figure 1.

In Figure 1, only 4 samples can be seen. As there are 12 samples for entire dataset, we conjecture that some samples overlap each other. It can be obviously concluded that much redundant information is contained in this dataset.

RS method is a well-established method for feature selection. It is widely used in many applications and more flexible than other methods, such as principal component analysis (PCA) [21]. One can use RS method to discover the data dependencies and reduce the number of attributes. For comparison purpose we used five other widely used feature selection methods (i) PCA, (ii) multidimensional scaling (MDS), (iii) generalized discriminant analysis (GDA), (iv) factor analysis (FA), and (v) Isomap.

TABLE 3: Discretization dataset.

| W1 | T1 | S | V | W2 | T2 | F | P | R |
|----|----|---|---|----|----|---|---|---|
| 1  | 0  | 1 | 0 | 1  | 0  | 1 | 1 | 0 |
| 1  | 0  | 0 | 1 | 0  | 1  | 1 | 0 | 1 |
| 0  | 1  | 1 | 0 | 0  | 1  | 0 | 1 | 0 |
| 0  | 1  | 1 | 0 | 1  | 0  | 0 | 1 | 0 |
| 1  | 0  | 0 | 1 | 0  | 1  | 0 | 1 | 0 |
| 1  | 0  | 0 | 1 | 1  | 0  | 0 | 1 | 0 |
| 1  | 0  | 1 | 0 | 0  | 1  | 1 | 0 | 1 |
| 0  | 1  | 1 | 0 | 0  | 1  | 1 | 0 | 0 |
| 1  | 0  | 0 | 1 | 1  | 0  | 0 | 1 | 1 |
| 0  | 1  | 0 | 1 | 1  | 0  | 0 | 1 | 0 |
| 1  | 0  | 0 | 1 | 1  | 0  | 0 | 1 | 1 |
| 0  | 1  | 0 | 1 | 1  | 0  | 0 | 1 | 0 |
| 1  | 0  | 1 | 0 | 1  | 0  | 0 | 1 | 1 |
| 0  | 1  | 1 | 0 | 1  | 0  | 1 | 1 | 0 |
| 1  | 0  | 1 | 0 | 1  | 0  | 1 | 1 | 0 |
| 0  | 1  | 1 | 0 | 1  | 0  | 1 | 1 | 0 |

TABLE 4: Reduced military simulation dataset.

| W1 | T1 | S | F | R |
|----|----|---|---|---|
| 1  | 0  | 1 | 0 | 1 |
| 0  | 1  | 1 | 0 | 1 |
| 1  | 0  | 0 | 1 | 1 |
| 1  | 0  | 1 | 0 | 1 |
| 1  | 0  | 0 | 1 | 1 |
| 1  | 0  | 1 | 0 | 1 |
| 1  | 0  | 0 | 1 | 0 |
| 0  | 1  | 1 | 0 | 0 |
| 1  | 0  | 0 | 1 | 0 |
| 0  | 1  | 1 | 0 | 0 |
| 0  | 1  | 0 | 1 | 0 |
| 0  | 1  | 1 | 0 | 0 |

After feature selection, more or less overlapping phenomenon appears in Figure 2, besides the ‘Rough set’ subfigure. The main goal of feature selection method is to choose a subset of useful variables, which exclude many redundant, but relevant variables. As seen in Figure 2, RS method obtains the most relevant variables, because ‘Rough set’ subfigure has the most overlapping region. The usefulness of these variables is directly judged by the estimated accuracy of the learning method. Learning method based on features chosen with RS method achieves error rates lower than other methods.

## 5. Performance Evaluation

In this section we report the performance of the proposed integrated method both on original and reduced military simulation datasets and compare the performance with other state-of-the-art classifiers. The reduced dataset is preprocessed using the procedure described in Section 4. After converting, 8 attributes in the original data are scattered to 16 attributes. In our classification application, the inputs (attributes) are normalized into the range  $[0, 1]$ , while the outputs (targets) are normalized into the range  $[-1, 1]$ . All the

simulations are running in the MATLAB R2008a (Windows version) environments with Intel 3.0 GHZ and 2 G RAM.

For comparison purpose we used two other widely used classification methods, ELM and SVM. For the implementation of SVM, we use the Spider library for MATLAB, which is publicly available from [22]. Both MATLAB codes of ELM and OELM can be downloaded from ELM host site: <http://www.ntu.edu.sg/home/egbhuang/>. To train SVM, radial basis function (RBF) kernel  $K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2)$  is used in our experiments. ELM with sigmoidal activation function  $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$  is tested. Both the Gaussian RBF activation function  $G(\mathbf{a}, b, \mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{a}\|^2/b)$  and the sigmoidal additive activation function have been used in the simulation of OELM. For both ELM and OELM with sigmoidal additive activation function and RBF activation function, the input weights  $\mathbf{a}$  and biases  $b$  are randomly generated from  $(-1, 1)^N \times (0, 1)$  based on the uniform probability distribution.

**5.1. Reduced Dataset.** According to the RS strategy in [4], the number of attributes is reduced from 16 to 8 on military simulation dataset, which is shown in Table 4.

**5.2. Selection of Parameters.** Two parameters for RBF kernel of SVM, the cost constant  $C$  and the kernel parameter  $\gamma$ , are both sensitive to the generalization performance. A straightforward way for model selection is the grid search strategy [23]. According to the method of [24], two parameters are tuned on a grid space of  $\{10^{-3}, 10^{-2}, 0.05, 10^{-1}, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 10^2, 10^3, 10^4\} \times \{10^{-3}, 10^{-2}, 10^{-1}, 0.2, 0.4, 0.8, 1, 2, 5, 10, 20, 50, 10^2, 10^3, 10^4\}$ . Therefore, for each problem we try  $15 \times 15 = 225$  combinations of parameters  $(C, \gamma)$  for SVM. Figure 3 shows the generalization performance of SVM for each combination.

As can be seen in Figure 3, the specific  $C$  and kernel parameter  $\gamma$  value form a grid in the  $(C, \gamma)$  parameter space. It is clear that the wider the search space is, the more possibility the grid search method finds the best parameter combination. However, the grid search method exhaustively searches the grid to find a best combination. Figure 3(a)

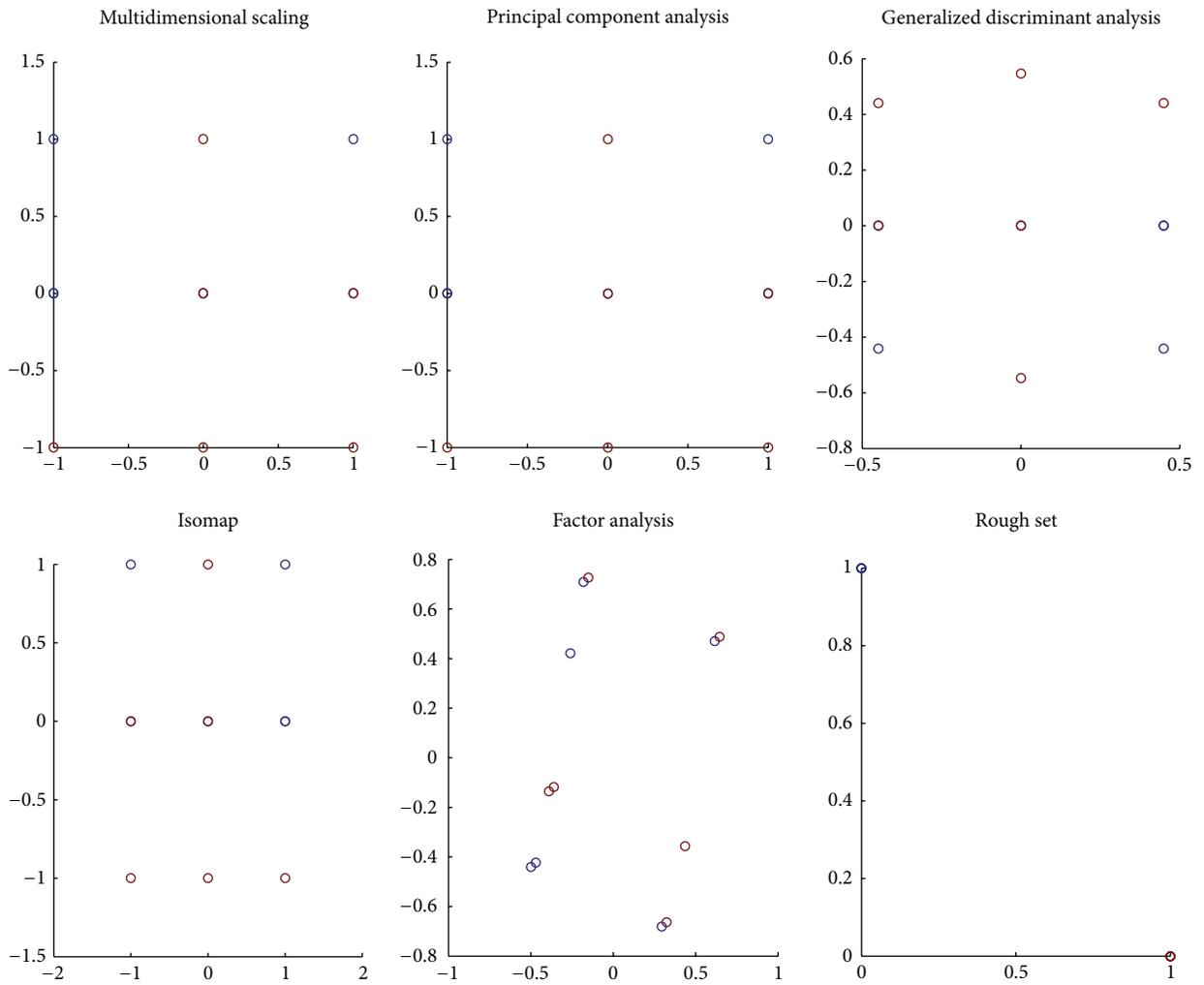


FIGURE 2: Comparison of 6 feature selection methods.

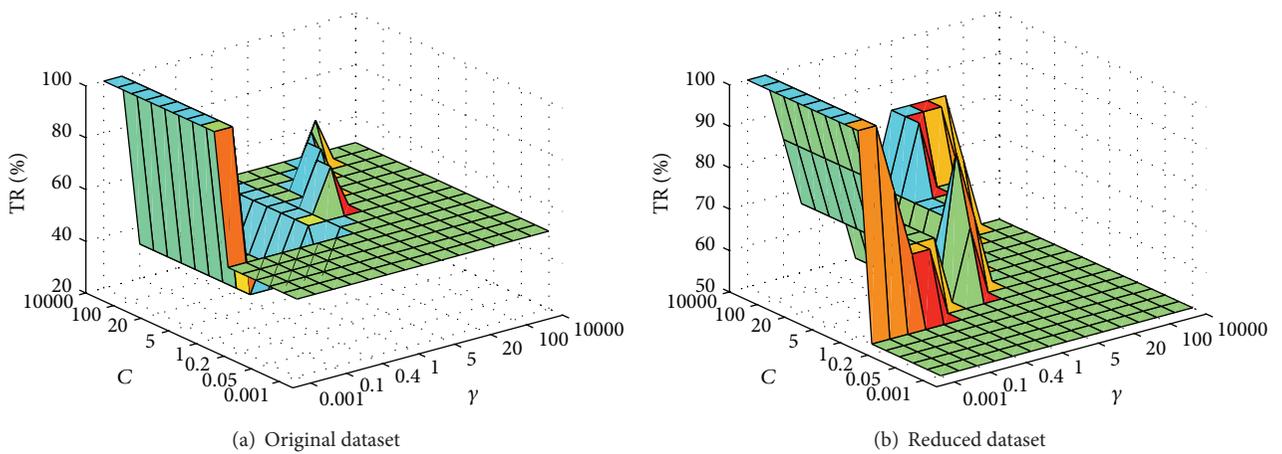


FIGURE 3: The performance of SVM for 225 combinations.

TABLE 5: Comparison between three ELM, SVM, and OELM for two datasets.

| Datasets         | Classifiers | $L/(L, C)/(C, \gamma)$ | Training time (s) | TR (%)       | Dev (%) |
|------------------|-------------|------------------------|-------------------|--------------|---------|
| Original dataset | ELM         | 5                      | $2.6243e - 004$   | 55.00        | 3.85    |
|                  | SVM         | $(1, 10^{-2})$         | 0.0079            | 59.33        | 2.67    |
|                  | OELM-S      | $(5, 10^2)$            | 0.0013            | <b>65.12</b> | 2.89    |
|                  | OELM-R      | $(5, 10^2)$            | $8.5874e - 004$   | 63.87        | 2.95    |
| Reduced dataset  | ELM         | 5                      | $2.7404e - 004$   | 60.33        | 2.79    |
|                  | SVM         | $(1, 10^{-2})$         | 0.0077            | 60.00        | 1.85    |
|                  | OELM-S      | $(5, 10^3)$            | 0.0018            | <b>74.67</b> | 2.06    |
|                  | OELM-R      | $(6, 10^2)$            | $9.8123e - 004$   | 72.92        | 2.36    |

TABLE 6: Random permutation performance comparison of ELM, SVM, and OELM-S.

| Trials | Random permutation<br>(training set) | TR (%)           |            |              |                 |            |              |
|--------|--------------------------------------|------------------|------------|--------------|-----------------|------------|--------------|
|        |                                      | Original dataset |            |              | Reduced dataset |            |              |
|        |                                      | ELM              | SVM        | OELM-S       | ELM             | SVM        | OELM-S       |
| 1      | (1, 8, 10, 2, 3, 12)                 | 50.00            | 33.33      | <b>66.67</b> | 66.67           | 33.33      | <b>83.33</b> |
| 2      | (3, 11, 6, 9, 10, 5)                 | 50.00            | 33.33      | 50.00        | 50.00           | 33.33      | 66.67        |
| 3      | (3, 9, 6, 11, 8, 10)                 | 83.33            | <b>100</b> | 83.33        | 83.33           | <b>100</b> | <b>100</b>   |
| 4      | (8, 3, 11, 1, 10, 5)                 | 50.00            | 33.33      | 50.00        | 66.67           | 33.33      | <b>83.33</b> |
| 5      | (9, 6, 3, 1, 8, 4)                   | 50.00            | <b>100</b> | 66.67        | 50.00           | <b>100</b> | 83.33        |
| 6      | (9, 8, 12, 7, 1, 6)                  | 50.00            | <b>100</b> | 66.67        | 66.67           | <b>100</b> | 83.33        |
| 7      | (5, 9, 7, 10, 3, 6)                  | 50.00            | 33.33      | 33.33        | 66.67           | 33.33      | <b>66.67</b> |
| 8      | (5, 2, 4, 10, 6, 8)                  | 33.33            | 16.67      | 16.67        | 33.33           | 16.67      | <b>50.00</b> |
| 9      | (3, 5, 11, 8, 10, 12)                | 66.67            | <b>100</b> | 66.67        | 83.33           | <b>100</b> | <b>100</b>   |
| 10     | (5, 2, 7, 12, 9, 1)                  | 50.00            | 33.33      | 50.00        | 50.00           | 33.33      | <b>66.67</b> |

indicates that the best generalization performance is achieved for 18 combinations on original dataset, and the performance of SVM highly depends on the combination of parameters. Similar results are obtained in Figure 3(b).

For ELM, there is only one parameter  $L$  (optimal number of hidden nodes) that needs to be determined. During our simulation it is found that ELM with 5 hidden nodes can generally obtain good performance for both datasets. Generally speaking, the generalization performance of OELM is not sensitive to the number of hidden nodes  $L$  [5]. We set the number of hidden nodes 1, 2, 3, 4, 5, and 6. Moreover, there is one more parameter that needs to be determined, which is the number of cost parameter  $C$ . Similar to [5], we set this parameter  $10^{-3}$ ,  $10^{-2}$ ,  $0.05$ ,  $10^{-1}$ ,  $0.2$ ,  $0.5$ ,  $1$ ,  $2$ ,  $5$ ,  $10$ ,  $20$ ,  $50$ ,  $10^2$ ,  $10^3$ , and  $10^4$ . The optimal parameter(s)  $L$  ( $L, C$ ) are selected to obtain the best testing performance. Table 5 shows the optimal parameters of three classifiers on two datasets.

**5.3. Performance Comparison on Two Datasets.** We evaluate the performance of three classifiers by repeated random splitting; that is, two datasets are equally partitioned into one training set and one test set. To avoid selection bias, there are at least one positive sample (labeled as “1”) and one negative sample (labeled as “-1”) in the training set. Such splitting is repeated 50 times. Training time, testing rate (TR) averaged over the 50 trials, and the corresponding standard deviations (Dev) are reported.

Table 5 draws a comparison between three classifiers, as well as some comparative results on different kernels. All of results represent the average, taken over 50 trials. OELM-S and OELM-R mean OELM using Sigmoidal and RBF kernel, respectively. As observed from Table 5, the performance of OELM-S and that of OELM-R is similar to each other except that OELM-S requires twice training time taken by OELM-R. We can see that the training time taken by ELM is much less than other classifiers, and the worst performance is also obtained by ELM. The performance of OELM-S or OELM-R is obviously higher than other classifiers. In order to further study this phenomenon, an extra experiment is conducted in Table 6.

Seen from Table 6, among 10 trials, the number of training and test set is fixed according to the above setting, but the order of training set is randomly shuffled for each trial. Although 100% testing rate is obtained for SVM on 4 trials, testing rate on other trials is no more than 33.33%. These results seem very unstable due to the randomness. Interestingly, OELM-S runs more stable than the other two classifiers, and this is why the performance of OELM-S is much higher than SVM and ELM.

Strictly speaking, it seems that, after data reduction, the performance of SVM cannot be improved on all trials. Moreover, OELM-S is exactly a good choice employed in our integrated strategy for improving the generalization performance of military simulation data.

## 6. Conclusions

This paper has introduced an integrated learning method of RS and OELM for military simulation dataset. The generalization performance of OELM is less sensitive to the user specified parameters especially for the number of hidden nodes. Compared to SVM, one can use OELM easily and effectively without parameter(s) tuning process. Significantly better results for the reduced dataset are obtained by employing OELM classifier over crisp discretization. This method has the potential for further military application because it can provide real-time decision support for commanders in battlefield.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## References

- [1] W. Ligu, X. Qing, and M. Xianquan, "The research of the data VV&C for the equipment war gaming based on the HLA," *Computer Engineering and Applications*, vol. 21, pp. 200–202, 2006.
- [2] M.-Z. Li, J.-K. Zhang, and W.-F. Che, "Research on data for combat simulation," *Command Control & Simulation*, vol. 32, no. 4, pp. 71–74, 2010.
- [3] H. Gao, H. Zhang, G. Chen et al., "Research and implementation of military simulation," *Fire Control & Command Control*, vol. 34, no. 2, pp. 150–153, 2009.
- [4] W.-M. Zhang and Q. Xue, "Application of rough set in date mining of warfare simulation," *Journal of System Simulation*, vol. 18, no. 2, pp. 179–181, 2006.
- [5] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1-3, pp. 155–163, 2010.
- [6] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, pp. 1651–1663, 2004.
- [7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.
- [8] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.
- [9] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, Hangzhou, China, June 2014.
- [10] J. W. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [11] J. W. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, pp. 66–77, 2012.
- [12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [13] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.
- [14] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko, "Rough sets," *Communications of the ACM*, vol. 38, no. 11, pp. 88–95, 1995.
- [15] Z. Pawlak and A. Skowron, "Rudiments of rough sets," *Information Sciences*, vol. 177, no. 1, pp. 3–27, 2007.
- [16] S. Greco, M. Benedetto, and R. Slowinski, "New developments in the rough set approach to multi-attribute decision analysis," *Bulletin of International Rough Set Society*, vol. 2, no. 2-3, pp. 57–87, 1998.
- [17] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Proceedings of the 12th International Conference on Machine Learning*, pp. 194–202, 1995.
- [18] R. Kerber, "Discretization of numeric attributes," in *Proceedings of the 10th National Conference on Artificial Intelligence*, pp. 123–128, MIT Press, Cambridge, Mass, USA, 1992.
- [19] F. Hussain, H. Liu, C. L. Tan, and M. Dash, "Discretization: an enabling technique," Tech. Rep., School of Computing, Singapore, 1999.
- [20] S. D. Bay, "Multivariate discretization of continuous variables for set mining," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 315–319, August 2000.
- [21] T. C. Lei, S. Wan, and T. Y. Chou, "The comparison of PCA and discrete rough set for feature extraction of remote sensing image classification—a case study on rice classification, Taiwan," *Computational Geosciences*, vol. 12, no. 1, pp. 1–14, 2008.
- [22] The Spider Library for MATLAB, <http://www.kyb.tuebingen.mpg.de/bs/people/spider/>.
- [23] N. Ancona, C. Cicirelli, E. Stella, and A. Distanto, "Object detection in images: run-time complexity and parameter selection of support vector machines," in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 2, pp. 426–429, August 2002.
- [24] P. Ghanty, S. Paul, and N. R. Pal, "NEUROSVM: an architecture to reduce the effect of the choice of kernel on the performance of SVM," *Journal of Machine Learning Research*, vol. 10, no. 3, pp. 591–622, 2009.

## Research Article

# Modeling and Optimization for Piercing Efficiency and Energy Consumption Based on Mean Value Staged KELM-PLS and GA Method

Dong Xiao<sup>1,2</sup> and Jichun Wang<sup>3</sup>

<sup>1</sup> State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110004, China

<sup>2</sup> Information Science and Engineering School, Northeastern University, Shenyang 110004, China

<sup>3</sup> College of Science, Liaoning Industry University, Jinzhou 121000, China

Correspondence should be addressed to Dong Xiao; [xiaodong@ise.neu.edu.cn](mailto:xiaodong@ise.neu.edu.cn)

Received 17 March 2014; Revised 22 May 2014; Accepted 23 May 2014; Published 17 June 2014

Academic Editor: Jiuwen Cao

Copyright © 2014 D. Xiao and J. Wang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Piercing manufacture of seamless tubes is the process that pierces solid blank into tube hollow. Piercing efficiency and energy consumption are the important indexes in the production of seamless tubes. Piercing process has the multivariate, nonlinear, cross-coupling characteristics. The complex factors that affect efficiency and consumption make it difficult to establish the mechanism models for optimization. Based on the production process, this paper divides the piercing process into three parts and proposes the piercing efficiency and energy consumption prediction models based on mean value staged KELM-PLS method. On the basis of mean value staged KELM-PLS prediction model, the minimum piercing energy consumption and maximum piercing efficiency are calculated by genetic optimization algorithm. Simulation and experiment prove that the optimization method based on the piercing efficiency and energy consumption prediction model can obtain the optimal process parameters effectively and also provide reliable evidences for practical production.

## 1. Introduction

In the production of seamless tube, the rotary reheating is the last working procedure of piercing production process. The tandem rolling production is the next working procedure of piercing production process. They both have higher production efficiency than piercing production process. So, the increase of production piercing efficiency plays a very important role in increasing the efficiency of overall production of seamless tube. Because the factors of affecting the piercing efficiency are rather complicated, it is difficult to build the mechanism model to find out its accurate value. Although the literature [1] analyzes the influence of parameters of piercing production process such as the rotational speed of guide disc and the rotational speed of roll, on the piercing efficiency, it only analyzes the influence of every parameter on the production efficiency statically and singly. Li [2] builds the model of relationship between the piercing efficiency and the wall thickness of shell. Although the literature [3, 4] analyzes

the influence of technological parameters such as the percent reduction of roll, the angle of inclination of roll, the rotational speed of roll, and the rotational speed of guide disc, on the piercing energy consumption production, it does not give the comprehensive relational model. The literature [5] analyzes the influence of shape of piercing head on the piercing energy consumption. The literature [6] introduces modeling and optimization for piercing energy consumption, but it did not make a comprehensive consideration for piercing energy consumption and piercing efficiency. The literature [7] introduces a modeling and optimization method for piercing energy consumption and piercing efficiency, but the model is linear which limits the accuracy of model prediction and the results of optimization. By the research for the production practice and the literature, it can be known that the main factors which affect the piercing efficiency include the rotational speed of roll, the percent reduction of roll, the rotational speed of guide disc, the size of tube blank and shell, the temperature of tube blank, and the feed angle of

roll. Because the state pays more and more close attention to the iron and steel industry with high energy consumption, every seamless tube manufacturer attaches more importance to the piercing energy consumption. Although to reduce the piercing energy consumption can meet the requirements of the direction of policy of the state and can also reduce the production cost of piercing, it may reduce the production efficiency. By the comprehensive consideration of all kinds of literatures and the actual production situations, this paper draws the following conclusion: the main production process parameters affecting the piercing energy consumption include the rotational speed, the percent reduction and the angle of inclination of roll, the size of tube blank and shell, the rotational speed of guide disc, and the temperature of tube blank.

In accordance with the characteristics of actual production of piercing process of tube blank, its working process can be divided into three substages: the primary unstable piercing (the first stage), the stable piercing (the second stage), and the secondary unstable piercing (the third stage), as shown in Figure 1. Through the analysis and comparison of the production process and data, it is found that the influences of different process parameter variables in the different piercing stages on the piercing efficiency and energy consumption are different. On this basis, we propose mean value substaged KELM-PLS method for the prediction of piercing efficiency and piercing energy consumption. This method selects the influence variable of process according to the piercing production stage, and the average value of corresponding variables in this stage is used as the input value. It overcomes the following shortcomings of traditional modeling method that the structure is complicated, the number of variables is larger, and the amount of calculation is considerable, and so forth. Finally, by the simulation inspection of the actual production data, the accuracy and rapidity of the introduced method are verified. After the modeling obtains the prediction model of accurate piercing efficiency and energy consumption, the optimum algorithm is utilized to carry out the comprehensive optimization of piercing efficiency and energy consumption on the basis of the model. This paper adopts the GA (genetic algorithm) [8–10]. The algorithm is a simple, fast, accurate, and effective global optimum algorithm. The algorithm can obtain the optimum production process parameters under specified conditions, which can be used to guide the actual production, in order to obtain the maximum economic benefit.

## 2. A Summary of Piercing Efficiency and Energy Consumption

The piercing efficiency of seamless tube is also called the axial sliding friction coefficient, which is the ratio of theoretical pure rolling time  $t_{th}$  to the actual pure rolling time  $t_{ac}$ , as shown in the formula,

$$y = \frac{t_{th}}{t_{ac}}. \quad (1)$$

In the formula,  $y$  is the piercing efficiency, %;  $t_{th}$  is the theoretical pure rolling time, s; and  $t_{ac}$  is the actual pure rolling time, s.

The main factors to affect the piercing efficiency include the rotational speed of roll, size, the shape and quality of material of tool, the size of tube blank, the size of shell, the feed angle, the temperature of tube blank, and the deformation system. The actual pure rolling time  $t_{ac}$  is measured directly by the field sensor, and the theoretical pure rolling time  $t_{th}$  can be expressed as the following formula:

$$t_{th} = \frac{l + L}{(\pi D_c n_r / 60) \sin \alpha}. \quad (2)$$

In the formula,  $n_r$  is the rotational speed of roll, r/min;  $\alpha$  is the feed angle, rad;  $D_1$  is the average diameter of roll, m;  $l$  is the length of deformed zone, m; and  $L$  is the length of shell, m. Because the manifestation of deformed zone is completely backward slip,  $t_{ac}$  is more than  $t_{th}$  all the time, and there is  $y \leq 1$  all the time.

The piercing energy consumption can be calculated by using the energy consumption produced in the course of piercing production of a steel tube. It can be found out by accumulating the electric energy consumed in producing this steel tube. The concrete expression is as the following:

$$E = \int_{t_0}^{t_1} P_{\text{piercing}} dt = \int_{t_0}^{t_1} U_{\text{piercing}} I_{\text{piercing}} dt. \quad (3)$$

In the formula,  $E$  is the piercing energy consumption, J;  $U_{\text{piercing}}$  is the operating voltage of piercing, V;  $I_{\text{piercing}}$  is the working current of piercing, A;  $t_0$  is the starting time of piercing, s; and  $t_1$  is the end time of piercing, s.

## 3. Analysis of Modeling Variables and Data Preprocessing

This paper firstly researches and analyzes the factor variables that affect the piercing efficiency and energy consumption. It is found in the research that some production variables affect the final piercing efficiency and energy consumption all the time and some production variables affect a part of the production stage only. The modeling variables of piercing efficiency and energy consumption obtained by the comprehensive comparison are shown in Table 1.

When building the model of efficiency and piercing energy consumption on the basis of mean value substaged KELM-PLS method, the average value of the modeling data is evaluated firstly. The average value of the data variables in three piercing stages is evaluated, respectively, and then the three-dimensional data is changed into the two-dimensional data. For the modeling data of piercing efficiency, the vector  $[x_{1,1}, x_{1,2}, \dots, x_{1,9}, x_{2,1}, x_{2,2}, \dots, x_{2,10}, x_{3,1}, x_{3,2}, \dots, x_{3,9}]$  formed by 28 variables can be obtained after it is changed into the two-dimensional data. For the modeling data of piercing energy consumption, the vector  $[x'_{1,1}, x'_{1,2}, \dots, x'_{1,6}, x'_{2,1}, x'_{2,2}, \dots, x'_{2,9}, x'_{3,1}, x'_{3,2}, \dots, x'_{3,8}]$  formed by 23 variables can be obtained after it is changed into the two-dimensional data. The data matrix  $X(I \times 28)$  of

TABLE 1: Modeling variable table for piercing efficiency and energy.

| Number | Original variables | The first stage variables |                    | The second stage variables |                    | The third stage variables |                    | The mean of variables                       |
|--------|--------------------|---------------------------|--------------------|----------------------------|--------------------|---------------------------|--------------------|---------------------------------------------|
|        |                    | Efficiency                | Energy consumption | Efficiency                 | Energy consumption | Efficiency                | Energy consumption |                                             |
| 1      | $x_1$              | $x_{1,1}$                 | $x_{1,1}$          | $x_{2,1}$                  | $x_{2,1}$          | $x_{3,1}$                 | $x_{3,1}$          | The percent reduction of upper roll         |
| 2      | $x_2$              | $x_{1,2}$                 | $x_{1,2}$          | $x_{2,2}$                  | $x_{2,2}$          | $x_{3,2}$                 | $x_{3,2}$          | The percent reduction of under roll         |
| 3      | $x_3$              | $x_{1,3}$                 | $x_{1,3}$          | $x_{2,3}$                  | $x_{2,3}$          | $x_{3,3}$                 | $x_{3,3}$          | The angle of inclination of upper roll      |
| 4      | $x_4$              | $x_{1,4}$                 | $x_{1,4}$          | $x_{2,4}$                  | $x_{2,4}$          | $x_{3,4}$                 | $x_{3,4}$          | The angle of inclination of under roll      |
| 5      | $x_5$              | $x_{1,5}$                 | $x_{1,5}$          | $x_{2,5}$                  | $x_{2,5}$          | $x_{3,5}$                 | $x_{3,5}$          | The rotational speed of upper roll          |
| 6      | $x_6$              | $x_{1,6}$                 | $x_{1,6}$          | $x_{2,6}$                  | $x_{2,6}$          | $x_{3,6}$                 | $x_{3,6}$          | The rotational speed of under roll          |
| 7      | $x_7$              | $x_{1,7}$                 |                    |                            |                    |                           |                    | The position change range of pusher         |
| 8      | $x_8$              | $x_{1,8}$                 |                    |                            |                    |                           |                    | The actual position of mandrel thrust block |
| 9      | $x_9$              | $x_{1,9}$                 |                    | $x_{2,7}$                  |                    | $x_{3,7}$                 |                    | The position of mandrel                     |
| 10     | $x_{10}$           |                           |                    | $x_{2,8}$                  | $x_{2,7}$          | $x_{3,8}$                 | $x_{3,7}$          | The rotational speed of left guide disc     |
| 11     | $x_{11}$           |                           |                    | $x_{2,9}$                  | $x_{2,8}$          | $x_{3,9}$                 | $x_{3,8}$          | The rotational speed of left guide disc     |
| 12     | $x_{12}$           |                           |                    | $x_{2,10}$                 | $x_{2,9}$          |                           |                    | The temperature of tube blank               |

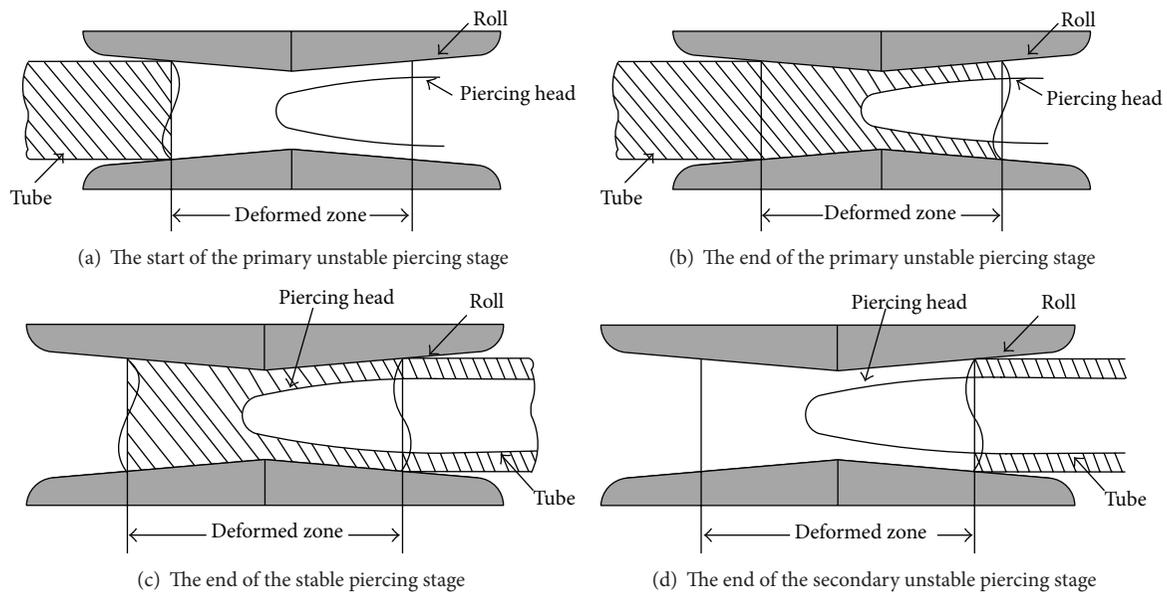


FIGURE 1: Definition of the parts for piercing process.

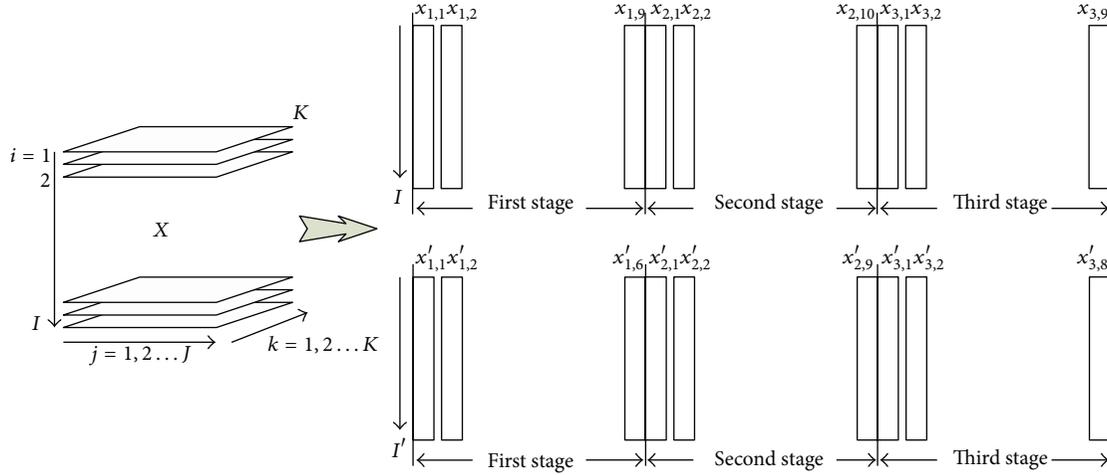


FIGURE 2: Unfolding of process data of three dimensions.

piercing efficiency and the data matrix  $X'(I' \times 23)$  of piercing energy consumption shown in Figure 2, which considers  $I$  production batches, are obtained.

#### 4. Building of Prediction Model Based on Mean Value Substaged KELM-PLS

**4.1. Nonlinear PLS.** Since linear PLS (partial least squares) model cannot describe correctly the nonlinear relation between independent variable  $X$  and dependent variable  $Y$ , nonlinear PLS method is required to solve this issue. Wold et al. extended the PLS method to nonlinear field [11, 12]. There are two feasible methods in nonlinear PLS methods: one is to perform array extension for input matrix, introduce some nonlinear terms of original variable, for example, the square term, and then regress the extended input and output matrix using PLS method. If prior knowledge on the relation of original input variable does not exist, this method cannot guide the selection of combined mode and may lead to oversized dimension of input matrix and the difficulties of processing; the other is to reserve the linear external model of PLS method. Internal model is nonlinear.

(1) External relation model is as follows:

$$\begin{aligned} X &= TP^T + E = \sum_{a=1}^A t_a p_a^T + E, \\ Y &= UQ^T + F = \sum_{a=1}^A u_a q_a^T + F, \end{aligned} \quad (4)$$

where  $A$  is the number of reserved eigenvectors,  $t_a(n \times 1)$  and  $u_a(n \times 1)$  are the score vectors of  $X$  and  $Y$ , respectively,  $p_a(m \times 1)$  and  $q_a(p \times 1)$  are the load vectors of  $X$  and  $Y$ , respectively,  $T(n \times A)$  and  $U(n \times A)$  are the score matrices of  $X$  and  $Y$ , respectively,  $P(m \times A)$  and  $Q(p \times A)$  are the load matrices of  $X$  and  $Y$ , respectively, and  $E$  and  $F$  are the fit residual matrices of  $X$  and  $Y$ , respectively.

(2) Internal relation model is as follows:

$$\hat{u}_a = f(t_a) + \varepsilon, \quad (5)$$

where  $f(\cdot)$  is the nonlinear function and  $\varepsilon$  is the residual.

The internal model of PLS method adopts neural network gains extensive application because neural network has the capability of fitting nonlinearity. As traditional feedforward neural network adopts gradient learning algorithm during training, parameters in network need iteration and update. Not only the training time lasts long but also it easily results in the issues of local minimum and excessive training [13].

**4.2. KELM Algorithm.** In supervised batch learning, the learning algorithms use a finite number of input-output samples for training [14–18]. For  $N$  arbitrary distinct samples  $(x_i, t_i) \in R^n \times R^m$ , where  $x_i$  is a  $n \times 1$  input vector and  $t_i$  is a  $m \times 1$  target vector, if an SLFN (single-hidden layer feedforward neural network [19–24]) with  $\tilde{N}$  hidden nodes can approximate these  $N$  samples with zero error, it then implies that there exists  $\beta_i, a_i$ , and  $b_i$ , such that

$$f_{\tilde{N}}(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i G(a_i, b_i, x_j) + \varepsilon_j = t_j, \quad (6)$$

where  $j = 1, \dots, N$ ,  $a_i$ , and  $b_i$  are the learning parameters of hidden nodes (weight vector connecting the input node to the hidden node and threshold of the hidden node) which are randomly selected according to the proof given by Huang et al. and  $\beta_i$  is the weight connecting the  $i$ th hidden node to the output node. To avoid overfitting the noise in the data, an error term  $\varepsilon_j$  is added.  $G(a_i, b_i, x)$  is the output of the  $i$ th hidden node with respect to the input  $x$  and  $\tilde{N}$  the number of hidden nodes which can be determined by trial and error or prior expertise. Then, equation can be written compactly as

$$H\beta = T, \quad (7)$$

where

$$H(a_1, \dots, a_{\bar{N}}, b_1, \dots, b_{\bar{N}}, x_1, \dots, x_N) = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_{\bar{N}}, b_{\bar{N}}, x_1) \\ \vdots & \cdots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_{\bar{N}}, b_{\bar{N}}, x_N) \end{bmatrix}_{N \times \bar{N}}, \quad (8)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\bar{N}}^T \end{bmatrix}_{\bar{N} \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m},$$

where  $H$  is called the hidden layer output matrix of the network; the  $i$ th column of  $H$  is the  $i$ th hidden node's output vector with respect to inputs  $x_1, x_2, \dots, x_N$ , and the  $j$ th row of  $H$  is the output vector of the hidden layer with respect to input  $x_j$ . The hidden node parameters  $a_i$  and  $b_i$  need not be tuned during training and may simply be assigned with random values. Equation (7) then becomes a linear system and the output weights  $\beta$  are estimated as

$$\tilde{\beta} = H^+ T, \quad (9)$$

where  $H^+$  is the Moore-Penrose generalized inverse of the hidden layer output matrix  $H$  [25–29].

Function  $g(\cdot)$  is usually unknown; we can incorporate kernel functions in  $g(\cdot)$  [30]. KELM (kernel extreme learning machine) can solve the problem of random initialization of ELM algorithm, and learning parameters of the model have a good Reuben [31]. It can improve the performance and accuracy of model prediction of piercing efficiency and piercing energy consumption. The kernel matrix  $K = [K(x; x_1), \dots, K(x; x_N)]^T$  ( $K(\cdot)$  is the kernel function) is introduced into (9) to estimate the output of the KELM:

$$o = KT. \quad (10)$$

Herein, the Gaussian kernel function (RBF) is adopted:

$$K(x_1; x_2) = \exp(-\lambda \|x_1 - x_2\|^2), \quad (11)$$

where  $\lambda$  needs to be specified.

**4.3. Mean Value Substaged KELM-PLS Modeling Steps.** The difference of nonlinear PLS modeling method based on KELM from linear PLS method is that it uses KELM to establish internal nonlinear model and in the meantime achieve the update of internal and external models. This method reserves linear external model, extracts through PLS the attributive information of process, eliminates the colinearity of data, reduces the dimension of input variable, adopts KELM to establish nonlinear internal model between input score vector matrix and output score vector, and raises the nonlinear processing capability of internal model. Thus, KELM-PLS method has the advantages of PLS and KELM, that is, the characteristics of robustness and feature extraction of PLS method and quick nonlinear processing capability of KELM.

The modeling and testing steps of nonlinear PLS method based on KELM are as follows.

(1) Assign two standardized data matrices,  $X \in R^{n \times m}$  and  $Y \in R^{n \times p}$ ; dynamic nonlinear PLS regression model can be expressed as follows:

$$X = [x_1, x_2, \dots, x_p]. \quad (12)$$

(2) Deploy the batch data of batch process, use cross-validation method to determine the number of latent variable, and adopt linear PLS method to calculate the score vector matrices  $T$  and  $U$  and load vector matrices  $P$  and  $Q$  for modeling sample  $X$  and  $Y$ .

Consider

$$X = TP^T + E = \sum_{a=1}^A t_a p_a^T + E, \quad (13)$$

$$Y = UQ^T + F = \sum_{a=1}^A u_a q_a^T + F.$$

(3) Assign the node number of ELM hidden layer and activation function (e.g., sigmoid function), use ELM to establish nonlinear model between internal models  $T$  and  $U$ , and gain  $U = f_{\text{KELM}}(T)$ , where  $f_{\text{KELM}}(\cdot)$  is the nonlinear function indicated by KELM. Hidden nodes in a SLFN transform feature space into another feature space. The original ELM defines the number of nodes as a parameter to be defined. We increase the number of hidden nodes until reaching a stop criteria (ex. residual error reduction), meanwhile, the number of hidden nodes is less than  $N$ .

(4) Use testing data to check model precision. Conduct PLS decomposition on the testing data  $X_2$ , gain score vector  $T_2$ :

$$X_2 = T_2 P^T + E. \quad (14)$$

Introduce  $T_2$  into KELM model, gain  $U_2 = f_{\text{KELM}}(T_2)$ , and find out model prediction value through  $\hat{Y} = UQ^T$ .

(5) The KELM-PLS method is used to build the prediction model of piercing efficiency and energy consumption for the data which is dealing with mean value substaged method according to Figure 2 and Table 1, and the model structure is shown in Figure 3.

The regression model between the data matrix  $X$  of piercing efficiency and the prediction value  $\hat{y}$  of piercing efficiency, which can be obtained by mean value substaged KELM-PLS method, is formula (15), and in the same way, the relational model between the data matrix  $X'$  of piercing energy consumption and the prediction value  $\hat{y}'$  of piercing

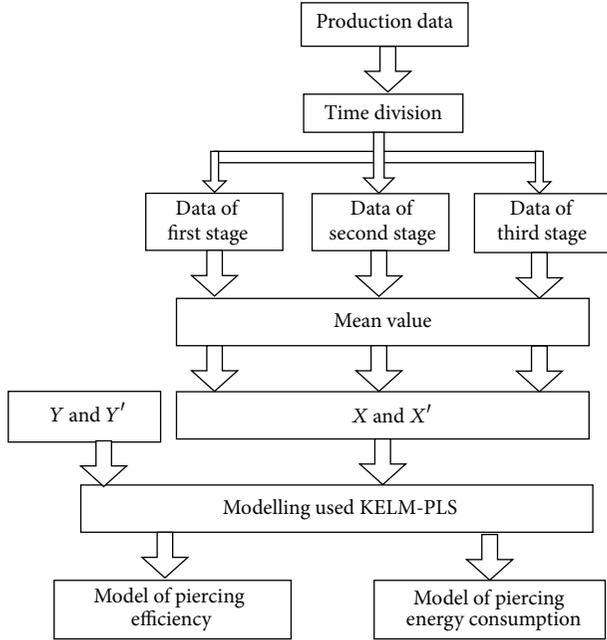


FIGURE 3: Chart of piercing efficiency and piercing energy model.

energy consumption, which can be obtained by substaged KELM-PLS method, is formula (16):

$$\begin{aligned} \hat{y} &= f(\mathbf{X}, \boldsymbol{\theta}) \\ &= f([x_{1,1}, x_{1,2}, \dots, x_{1,9}, x_{2,1}, x_{2,2}, \dots, x_{2,10}, \\ &\quad x_{3,1}, x_{3,2}, \dots, x_{3,9}], \boldsymbol{\theta}), \end{aligned} \quad (15)$$

$$\begin{aligned} \hat{y}' &= f(\mathbf{X}', \boldsymbol{\theta}') \\ &= f([x'_{1,1}, x'_{1,2}, \dots, x'_{1,6}, x'_{2,1}, x'_{2,2}, \dots, x'_{2,9}, \\ &\quad x'_{3,1}, x'_{3,2}, \dots, x'_{3,8}], \boldsymbol{\theta}'). \end{aligned} \quad (16)$$

Among them,  $\boldsymbol{\theta}$  and  $\boldsymbol{\theta}'$  are the regression coefficients of model of piercing efficiency and energy consumption obtained by utilizing mean value substaged KELM-PLS.

## 5. Comprehensive Optimization of Piercing Efficiency and Energy Consumption on the Basis of Genetic Algorithm

After building the accurate model, we put the model into the objective function and utilize the genetic algorithm to optimize and find out the optimal production process parameters according to the different conditions of market and production, in order to guarantee the maximum profit of enterprise.

**5.1. Decision Variable and Optimization Objective of Piercing Efficiency and Energy Consumption.** For the upper roll and lower roll and left guide disc and right guide disc, although their real time parameters of actual production are different,

the setting values to be controlled are set to the same setting values, and, therefore, they have merged into one production parameter in optimization when carrying out the optimization. As shown in Table 2, the decision variables related to the piercing efficiency are  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{17}$ , and the decision variables related to the piercing energy consumption are  $\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_7, \bar{x}_8, \bar{x}_9, \bar{x}_{10}, \bar{x}_{12}, \bar{x}_{13}, \bar{x}_{14}, \bar{x}_{15}, \bar{x}_{17}$ . Their constraint conditions are formulas (18)~(22) and the concrete change range is as follows: the temperature of tube blank: 1200~1300°C, the rotational speed of guide disc: 20~30 r/min, the actual position of mandrel thrust block: 115~170 mm, the position change range of pusher: 0~2550 mm, the position of mandrel: 200~300 mm, the rotational speed of roll: 105~195 r/min, the angle of inclination of roll: 10~14° mm, and the percent reduction of roll: 120~170 mm. In order to obtain the maximum profit of production, it is necessary to carry out the comprehensive optimization of piercing efficiency and energy consumption and take the following formula as the optimization model to seek the optimum decision variable value of  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{17}$ :

$$\max y = k_1 f(\mathbf{x}, \boldsymbol{\theta}) + \frac{k_2}{f(\mathbf{x}', \boldsymbol{\theta}')}, \quad (17)$$

where  $f(\mathbf{x}, \boldsymbol{\theta})$  and  $f(\mathbf{x}', \boldsymbol{\theta}')$  are piercing efficiency and energy consumption, respectively, which are calculated by mean value substaged KELM-PLS method;  $k_1$  and  $k_2$  are the weighted values of adjustment of optimization objective, the adjustment range of  $k_1$  is 0~13, and the adjustment range of  $k_2$  is 0~11. When the rate of capacity utilization is lower, it is necessary to maximize  $k_2$ , and in this way the energy consumption in production can be reduced and at the same time the production cost can be reduced, thus maximizing the production benefit. Similarly, when the rate of capacity utilization is higher and the supply and marketing of product are flourishing, it is necessary to maximize  $k_1$ , in order to guarantee that the production line of seamless tube operates in the state of maximum efficiency, and in this way, the maximum benefit of production can also be obtained. In other conditions, it is necessary to carry out the adjustment according to the experience and concrete conditions.

The constraint condition of production optimization is as follows:

(1) the change range of production process parameters of piercing equipment

$$\begin{aligned} \text{s.t.1} \quad & 105 \leq \bar{x}_1, \bar{x}_8, \bar{x}_{13} \leq 195, \\ & 0 \leq \bar{x}_4 \leq 2550, \\ & 115 \leq \bar{x}_5 \leq 170, \\ & 1200 \leq \bar{x}_7 \leq 1300, \\ & 200 \leq \bar{x}_6, \bar{x}_{11}, \bar{x}_{16} \leq 300, \\ & 120 \leq \bar{x}_2, \bar{x}_9, \bar{x}_{14} \leq 170, \\ & 10 \leq \bar{x}_3, \bar{x}_{10}, \bar{x}_{15} \leq 14, \\ & 20 \leq \bar{x}_{12}, \bar{x}_{17} \leq 30; \end{aligned} \quad (18)$$

TABLE 2: Decision making variable table of optimization.

| Number | Variables      | Decision variables of piercing efficiency | Decision variables of piercing energy consumption | The mean of variables                              |
|--------|----------------|-------------------------------------------|---------------------------------------------------|----------------------------------------------------|
| 1      | $\bar{x}_1$    | √                                         | √                                                 | The rotational speed of roll at first stage        |
| 2      | $\bar{x}_2$    | √                                         | √                                                 | The percent reduction of roll at first stage       |
| 3      | $\bar{x}_3$    | √                                         | √                                                 | The angle of inclination of roll at first stage    |
| 4      | $\bar{x}_4$    | √                                         |                                                   | The position change range of pusher                |
| 5      | $\bar{x}_5$    | √                                         |                                                   | The actual position of mandrel thrust block        |
| 6      | $\bar{x}_6$    | √                                         |                                                   | The position of mandrel at first stage             |
| 7      | $\bar{x}_7$    | √                                         | √                                                 | The temperature of tube blank at second stage      |
| 8      | $\bar{x}_8$    | √                                         | √                                                 | The rotational speed of roll at second stage       |
| 9      | $\bar{x}_9$    | √                                         | √                                                 | The percent reduction of roll at second stage      |
| 10     | $\bar{x}_{10}$ | √                                         | √                                                 | The angle of inclination of roll at second stage   |
| 11     | $\bar{x}_{11}$ | √                                         |                                                   | The position of mandrel at second stage            |
| 12     | $\bar{x}_{12}$ | √                                         | √                                                 | The rotational speed of guide disc at second stage |
| 13     | $\bar{x}_{13}$ | √                                         | √                                                 | The rotational speed of roll at third stage        |
| 14     | $\bar{x}_{14}$ | √                                         | √                                                 | The percent reduction of roll at third stage       |
| 15     | $\bar{x}_{15}$ | √                                         | √                                                 | The angle of inclination of roll at third stage    |
| 16     | $\bar{x}_{16}$ | √                                         |                                                   | The position of mandrel at third stage             |
| 17     | $\bar{x}_{17}$ | √                                         | √                                                 | The rotational speed of guide disc at third stage  |

TABLE 3: Comparison of the production index after optimization of piercing efficiency and energy consumption.

| Method of modeling            | Piercing efficiency |                  | Piercing energy consumption |                  |
|-------------------------------|---------------------|------------------|-----------------------------|------------------|
|                               | Accuracy of model   | Time of modeling | Accuracy of model           | Time of modeling |
| Mean value substaged KELM-PLS | 92.14               | 0.39             | 93.14                       | 0.32             |
| MICR                          | 91.12               | 0.36             | 90.53                       | 0.30             |
| MPLS                          | 90.21               | 0.52             | 89.56                       | 0.48             |

TABLE 4: Optimization result of piercing efficiency and energy consumption.

| Number | Decision variables | The mean of variables                              | Optimization results in the highest efficiency piercing | Optimization results in the lowest energy consumption | Unit  |
|--------|--------------------|----------------------------------------------------|---------------------------------------------------------|-------------------------------------------------------|-------|
| 1      | $\bar{x}_1$        | The rotational speed of roll at first stage        | 136.72                                                  | 120.91                                                | r/min |
| 2      | $\bar{x}_2$        | The percent reduction of roll at first stage       | 144.75                                                  | 140.13                                                | mm    |
| 3      | $\bar{x}_3$        | The angle of inclination of roll at first stage    | 13.71                                                   | 12.33                                                 | °     |
| 4      | $\bar{x}_4$        | The position change range of pusher                | 2549.63                                                 | 2547.32                                               | mm    |
| 5      | $\bar{x}_5$        | The actual position of mandrel thrust block        | 115.64                                                  | 112.42                                                | mm    |
| 6      | $\bar{x}_6$        | The position of mandrel at first stage             | 279.32                                                  | 279.43                                                | mm    |
| 7      | $\bar{x}_7$        | The temperature of tube blank at second stage      | 1286.23                                                 | 1251.54                                               | °C    |
| 8      | $\bar{x}_8$        | The rotational speed of roll at second stage       | 171.82                                                  | 154.43                                                | °     |
| 9      | $\bar{x}_9$        | The percent reduction of roll at second stage      | 144.86                                                  | 142.75                                                | mm    |
| 10     | $\bar{x}_{10}$     | The angle of inclination of roll at second stage   | 13.47                                                   | 12.49                                                 | °     |
| 11     | $\bar{x}_{11}$     | The position of mandrel at second stage            | 280.56                                                  | 277.78                                                | mm    |
| 12     | $\bar{x}_{12}$     | The rotational speed of guide disc at second stage | 26.74                                                   | 26.15                                                 | r/min |
| 13     | $\bar{x}_{13}$     | The rotational speed of roll at third stage        | 152.51                                                  | 137.69                                                | r/min |
| 14     | $\bar{x}_{14}$     | The percent reduction of roll at third stage       | 146.43                                                  | 142.74                                                | mm    |
| 15     | $\bar{x}_{15}$     | The angle of inclination of roll at third stage    | 13.73                                                   | 12.34                                                 | °     |
| 16     | $\bar{x}_{16}$     | The position of mandrel at third stage             | 280.64                                                  | 278.87                                                | mm    |
| 17     | $\bar{x}_{17}$     | The rotational speed of guide disc at third stage  | 28.65                                                   | 26.78                                                 | r/min |

(2) the bite condition of seamless tube piercing production

$$\text{s.t.2 } 2T_i > Q_i + 2P_i. \quad (19)$$

Among them,  $i$  is the number of substage of piercings,  $i = 1, 2, 3$ ;  $T_i$  is the axial dragged-into frictional force  $T_i = p_i f b_i L_{Hi} \cos \theta_i$  of one roll to affect the tube blank,  $N$ ;  $P_i$  is the positive pressure axial component  $P_i = p_i b_i L_{Hi} \sin \theta_i$  of one roll to affect the tube blank,  $N$ ; and  $Q_i$  is the resistance of piercing head,  $N$ .

Consider

$$Q_i = p_i \pi r_H^2 \times \left( \frac{2\pi n_i D_G}{60 d_{GJ}} \sin \theta_i + \frac{1 + (d_{KJ}/d_{GJ})^2}{1 + d_{KJ}/d_{GJ}} \right) \left( \frac{2\pi n_i}{60} \right)^{-1}. \quad (20)$$

In the formula,  $p_i$  is the mean specific roll pressure,  $\text{Kg/mm}^2$ ;  $f$  is the friction coefficient;  $\theta_i$  is the angle of inclination of roll, rad;  $n_i$  is the rotational speed of roll, rad/min;  $r_H$  is the radius of nose of piercing head, mm;  $L_{Hi}$  is the length of deformed zone, mm;  $b_i$  is the percent reduction of roll, mm;  $D_G$  is the diameter of roll waist, mm;  $d_{GJ}$  is the distance of roll at the place of inlet, mm; and  $d_{KJ}$  is the distance of roll at the place of bore throat, mm.

In the constrained conditions shown in formula (17), the rotational speed of rolls  $n_1, n_2, n_3$  corresponds to the decision variables  $\bar{x}_1, \bar{x}_8, \bar{x}_{13}$ , respectively; the percent reduction of roll  $b_1, b_2, b_3$  corresponds to the decision variables  $\bar{x}_2, \bar{x}_9, \bar{x}_{14}$ , respectively; the angle of inclination of rolls  $\theta_1, \theta_2, \theta_3$  corresponds to the decision variables  $\bar{x}_3, \bar{x}_{10}, \bar{x}_{15}$ , respectively; and the position of mandrels  $L_{H1}, L_{H2}, L_{H3}$  corresponds to the decision variables  $\bar{x}_6, \bar{x}_{11}, \bar{x}_{16}$ , respectively.

(3) The quality of shell produced by the cross piercing should meet the requirements of site

$$\text{s.t.3 } M_P = f_{\text{quality}}(x) < M_N. \quad (21)$$

In the formula,  $M_N$  = the requirements for quality, %;  $M_P$  = the quality of shell, %; and  $f_{\text{quality}}()$  = the quality of shell calculated by the quality prediction model [32].

Finally, by formula (17)~formula (22), the model of comprehensive optimization of piercing efficiency and energy consumption can be expressed as follows:

$$\begin{aligned} \max \quad & y = k_1 \mathbf{x}\boldsymbol{\theta} + k_2 \mathbf{x}'\boldsymbol{\theta}' \\ \text{s.t.} \quad & 105 \leq \bar{x}_1, \bar{x}_8, \bar{x}_{13} \leq 195, \\ & 0 \leq \bar{x}_4 \leq 2550, \\ & 115 \leq \bar{x}_5 \leq 170, \\ & 1200 \leq \bar{x}_7 \leq 1300, \\ & 200 \leq \bar{x}_6, \bar{x}_{11}, \bar{x}_{16} \leq 300, \end{aligned}$$

$$120 \leq \bar{x}_2, \bar{x}_9, \bar{x}_{14} \leq 170,$$

$$10 \leq \bar{x}_3, \bar{x}_{10}, \bar{x}_{15} \leq 14,$$

$$20 \leq \bar{x}_{12}, \bar{x}_{17} \leq 30, \quad (22)$$

$$2T_i > Q_i + 2P_i,$$

$$M_P = f_{\text{quality}}(x) < M_N.$$

**5.2. Comprehensive Optimization of Piercing Efficiency and Energy Consumption.** The comprehensive optimization of piercing efficiency and energy consumption is an optimization issue with constraint condition, and this paper selects the genetic algorithm as the algorithm to find the solution of model of comprehensive optimization of piercing efficiency and energy consumption. Flow chart of piercing efficiency and energy optimization is shown in Figure 4.

(1) The definition of the fitness function of piercing efficiency and energy consumption optimization is as follows:

$$\max \quad y = k_1 f(\mathbf{x}, \boldsymbol{\theta}) + \frac{k_2}{f(\mathbf{x}', \boldsymbol{\theta}')}. \quad (23)$$

In the formula,  $\boldsymbol{\theta}$  is the coefficient of model of piercing efficiency;  $\boldsymbol{\theta}'$  is the coefficient of model of piercing energy consumption;  $\mathbf{x}$  is the decision variable of optimization of piercing efficiency; and  $\mathbf{x}'$  is the decision variable of optimization of piercing energy consumption.  $f()$  is the model output of mean value substaged KELM-PLS method.

(2) Because the comprehensive optimization of piercing efficiency and energy consumption is an issue of continuous parameters optimization, float encoding is adopted as the encoding mode. It avoids the length limitation of binary encoding which reduces the performance and solution accuracy. Float encoding does not require coding and decoding operation which improves the computing speed and accuracy to solve. At the same time, the integral arithmetic crossover algorithm is adopted [33, 34].

(3) Punishment technology is commonly used for constrained optimization problems in genetic algorithm. Penalty function methods transform a constrained optimization problem into a sequence of unconstrained optimization problems. The constraints are appended to the objective function via a penalty parameter and a penalty function. In general, a feasible penalty function should admit a positive penalty for infeasible points and no penalty for feasible points. The fitness function is designed for

$$F = y - m_1 \times \max\{0, g_1(x)\} - m_2 \times \max\{0, g_2(x)\} - m_3 \times \max\{0, g_3(x)\} - m_4 h(x), \quad (24)$$

where  $g_1$  is the inequality constraint of formula (18),  $g_2$  is the inequality constraint of formula (19),  $g_3$  is the inequality constraint of formula (21), and  $h$  is the equality constraint of formula (20).

(4) GA algorithm stopping convergence condition is as follows. When the algorithm is run continuously for 20 generations, it stops if the fitness value changes to less than  $10^{-6}$ . And it stops when the algorithm is run continuously for 1000 generations.

### 6. Simulation and Experiment

In order to verify the accuracy of the method, this paper selects the data of production of 90 piercing tubes of Diescher Mannesmann piercer of seamless tube factory of Baosteel company in January 2014. The first 65 shells are used to build the prediction model of piercing efficiency and energy consumption and the last 25 shells are used for the inspection of accuracy of model. The test process conditions are as follows: the diameter of roll:  $D_g = 1000$  mm, the cone angle of inlet:  $\beta = 2.75^\circ$ , the diameter of guide disc:  $D_p = 1800$  mm, the toe angle:  $\Phi = 0^\circ$ , the distance of guide disc:  $L_p = 173$  mm, the rotational speed of guide disc: 26 revolutions per minute, the initial temperature of guide disc:  $100^\circ\text{C}$ , the rotational speed of roll: 185 revolutions per minute, the initial temperature of roll:  $150^\circ\text{C}$ , the percent reduction of roll: 150 mm, the change range of angle of inclination of roll:  $13^\circ$ , the temperature of piercing head:  $100^\circ\text{C}$ , the temperature of tube blank:  $1250^\circ\text{C}$ , the specification of tube blank:  $\Phi 178$  mm, the specification of shell:  $\Phi 179$  mm  $\times$  30.25 mm, and the quality of material: C22 (20# steel).

First, the production data of 90 shells are handled to obtain the input data  $\mathbf{X}(65, 28)$  and  $\mathbf{X}'(65, 23)$  for modeling of piercing efficiency and energy consumption as well as the input data  $\mathbf{X}_{\text{new}}(25, 28)$  and  $\mathbf{X}'_{\text{new}}(25, 23)$  for inspection of piercing efficiency and energy consumption according to Figure 2 and Table 1. After using the mean value sub-staged KELM-PLS method to build the prediction models of piercing efficiency and energy consumption, respectively, the input data  $\mathbf{X}_{\text{new}}(25, 28)$  and  $\mathbf{X}'_{\text{new}}(25, 23)$  for inspection of piercing efficiency and energy consumption are put into formula (5), respectively, and their precision of model is compared with the precision of the model of actual values  $\mathbf{y}(25, 1)$  and  $\hat{\mathbf{y}}'(25, 1)$ . And at the same time, the prediction model of efficiency and piercing energy consumption of traditional MPLS method is built. The following 12 process variables that affect the piercing efficiency are selected: the intermesh of upper roll and lower roll, the values of angles of inclination of upper and lower rolls, the rotational speed of upper roll and lower roll, the position of pusher, the actual position of mandrel thrust block, the position of mandrel, the rotational speed of left and right discs, and the temperature of tube blank. Similarly, the production data of 25 shells are utilized to carry out the inspection of model. Figures 5 and 6 show the results of comparison among the inspections of the models of piercing efficiency and the piercing energy consumption of three kinds of methods which are mean value sub-staged KELM-PLS, MICR [7], and MPLS method.

As shown in Figures 5 and 6, the prediction accuracies of model of piercing efficiency on the basis of mean value sub-staged KELM-PLS, MICR, and MPLS model are 92.14%, 91.12, and 90.21%, respectively, and the input variables of their models are 28 and 1101 variables, respectively. The calculation times used by the models are 0.39 seconds, 0.36 seconds, and 0.52 seconds. The prediction accuracies of model of piercing energy consumption on the basis of mean value sub-staged KELM-PLS, MICR, and MPLS model are 93.14%, 90.53%, and

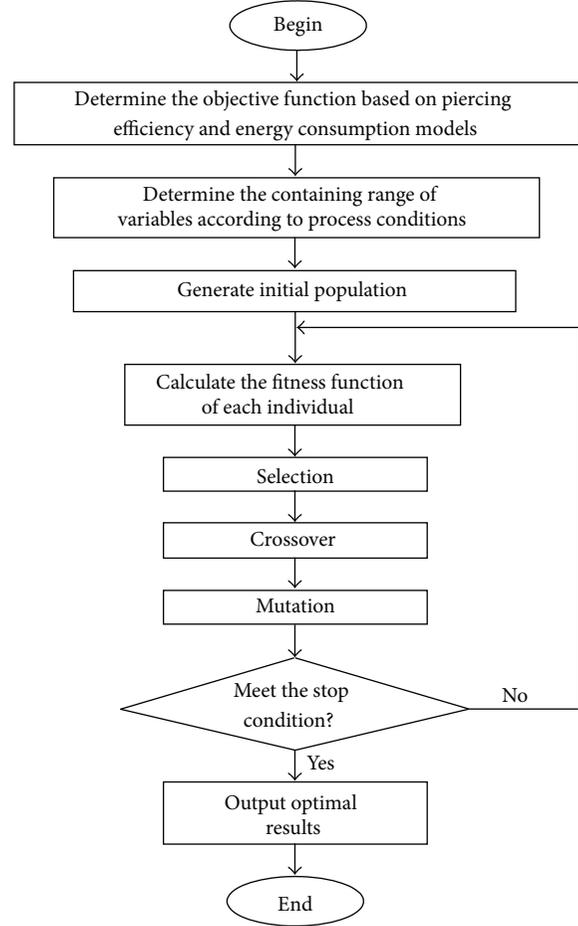


FIGURE 4: Flow chart of optimization.

89.56%, respectively, and the input variables of their models are 23 and 801 variables, respectively. The calculation times of models are 0.32 seconds, 0.30 seconds, and 0.48 seconds, respectively. Table 3 indicates the comparison among the three kinds of methods of piercing efficiency and energy consumption. It can be found from this condition that the model of mean value sub-staged KELM-PLS method increases the prediction accuracy of model to a certain extent, and at the same time, the reduction of input quantity of model also greatly reduces the calculation time used by the model, which makes the model more easy to use online. Figure 7 shows that maximum piercing efficiency is taken as the strategy of seeking optimization when the influence of the piercing energy consumption is not considered. The maximum piercing efficiency obtained by the optimization is 90.24%. Figure 8 shows that the minimum energy piercing consumption is taken as the strategy of seeking optimization when the piercing efficiency is not considered. The minimum piercing energy consumption obtained by the optimization is 8.268 Kwh. And at the same time, the production process parameter values in every stage that minimize the piercing energy consumption are found out. Table 4 indicates the parameter values of production process in every stage and Table 5 indicates the comparison between the indexes of

TABLE 5: Comparison of the production index after optimization of piercing efficiency and energy consumption.

| Production index of piercing | Before optimization | After optimization with MICR model | After optimization with KELM-PLS model | Unit |
|------------------------------|---------------------|------------------------------------|----------------------------------------|------|
| Piercing efficiency          | 84.56               | 88.93                              | 90.24                                  | %    |
| Piercing energy consumption  | 11.352              | 8.341                              | 8.268                                  | KWh  |

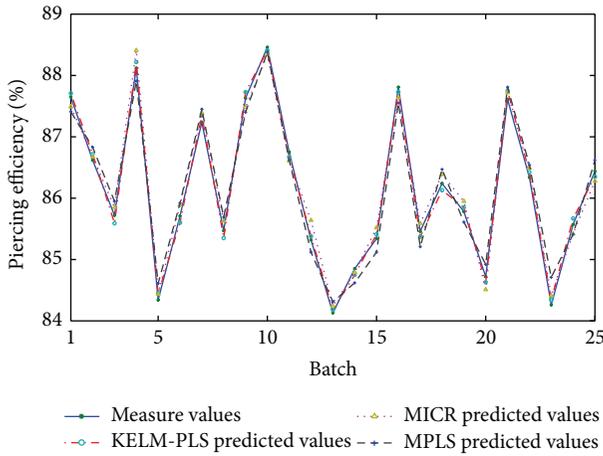


FIGURE 5: Test result of piercing efficiency model.

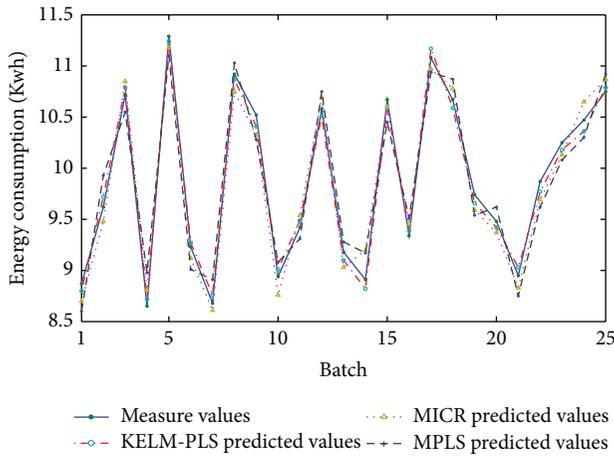


FIGURE 6: Test result of piercing energy consumption model.

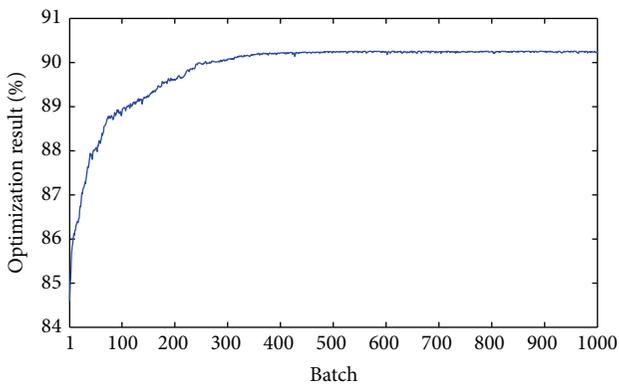


FIGURE 7: Optimization result with maximum piercing efficiency.

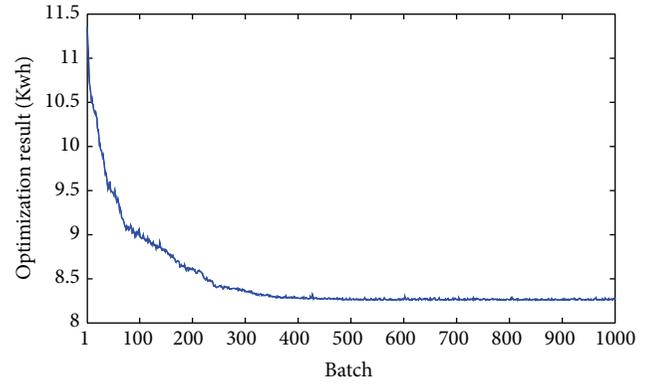


FIGURE 8: Optimization result with minimum piercing energy consumption.

piercing efficiency and energy consumption before and after the optimization.

### 7. Conclusions

For the process of producing the shell of seamless tube by piercing, this paper adopts the mean value substaged KELM-PLS modeling method, builds the prediction model of piercing efficiency and energy consumption, and provides the online prediction of piercing efficiency and energy consumption of tube blank. On this basis, the genetic algorithm is adopted for the comprehensive optimization of piercing efficiency and energy consumption and the optimum production process parameters are obtained according to the different market demands and constraint conditions. The test indicates that when the minimum energy consumption is required, the optimized energy consumption of production is reduced obviously, and at the same time, the cost of production is reduced, and when the high production efficiency is required, the optimized production increases the piercing efficiency to a certain extent, which can guarantee that the production task is finished smoothly. Meanwhile, the method can also be spread to the application to the index predication and optimization of other multistage batch processes.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant nos. 61203214), National Natural Science Foundation of China (Grant nos. 61203103, 61374146, 61374147) and Provincial Science and Technology Department of Education Projects (L2013101).

## References

- [1] M. A. Cavaliere, M. B. Goldschmit, and E. N. Dvorkin, "Finite element analysis of steel rolling processes," *Computers and Structures*, vol. 79, no. 22-25, pp. 2075–2089, 2001.
- [2] L.-S. Li, *Steel Tube Plastic Deformation Theory*, Metallurgical Industry Press, Beijing, China, 1985, (Chinese).
- [3] Y.-H. Shuang, *Numerical Simulation Theory and Cross Rolling Steel Production Process*, Metallurgical Industry Press, Beijing, China, 2001, (Chinese).
- [4] X.-Y. Yang and Y. Liu, "Establishment of math model for piercing efficiency," *Steel Tube*, vol. 24, no. 5, pp. 30–33, 1995 (Chinese).
- [5] Y.-H. Pang, Z.-M. Zhou, and Y.-Y. Cheng, "Influence of plug configuration on wall-thickness uniformity and energy consumption during shell piercing," *Steel Tube*, vol. 20, no. 5, pp. 48–53, 1991 (Chinese).
- [6] D. Xiao, X.-L. Pan, Y. Yuan, Z.-Z. Mao, and F.-L. Wang, "Modeling and optimization for piercing energy consumption," *Journal of Iron and Steel Research International*, vol. 16, no. 2, pp. 40–44, 2009.
- [7] D. Xiao, J.-C. Wang, X.-L. Pan, and Z.-Z. Mao, "Modeling and optimization for piercing efficiency and energy consumption," *Journal of Iron and Steel Research*, vol. 25, no. 1, pp. 14–22, 2013 (Chinese).
- [8] W. Kitagawa, Y. Kimura, and T. Takeshita, "A study on embedding genetic algorithm to three-dimensional finite element method by using shell script," *IEEJ Transactions on Industry Applications*, vol. 132, no. 2, pp. 227–232, 2012.
- [9] D.-L. Liu, X.-H. Chen, and J.-L. Du, "A hybrid genetic algorithm for constrained optimization problems," *Journal of Computers*, vol. 8, no. 2, pp. 272–278, 2013.
- [10] S.-F. Ding, L. Xu, C.-Y. Su, and F.-X. Jin, "An optimizing method of RBF neural network based on genetic algorithm," *Neural Computing and Applications*, vol. 21, no. 2, pp. 333–336, 2012.
- [11] S. Wold, N. Kettaneh-Wold, and B. Skagerberg, "Nonlinear PLS modeling," *Chemometrics and Intelligent Laboratory Systems*, vol. 7, no. 1-2, pp. 53–65, 1989.
- [12] S. J. Qin, "Recursive PLS algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, 1998.
- [13] B. Hu, Z. Zhao, and J. Liang, "Multi-loop nonlinear internal model controller design under nonlinear dynamic PLS framework using ARX-neural network model," *Journal of Process Control*, vol. 22, no. 1, pp. 207–217, 2012.
- [14] A. Bueno-Crespo, P. J. García-Laencina, and J.-L. Sancho-Gómez, "Neural architecture design based on extreme learning machine," *Neural Networks*, vol. 48, pp. 19–24, 2013.
- [15] W. Xi-Zhao, S. Qing-Yan, M. Qing, and Z. Jun-Hai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3–9, 2013.
- [16] J. Zhai, H. Xu, and Y. Li, "Fusion of extreme learning machine with fuzzy integral," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 21, supplement 2, pp. 23–34, 2013.
- [17] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, 2014.
- [18] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, 2014.
- [19] J. W. Cao, T. Chen, and J. Fan, "Fast online learning algorithm for landmark recognition based on BoW framework," in *Proceedings of the 9th IEEE Conference on Industrial Electronics and Applications*, June 2014.
- [20] Y. Jin, J. W. Cao, Q. Q. Ruan, and X.-Q. Wang, "Cross-modality 2D-3D face recognition via multiview smooth discriminant analysis based on ELM," *Journal of Electrical and Computer Engineering*, vol. 2014, Article ID 584241, 9 pages, 2014.
- [21] G.-B. Huang, D.-H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [22] G. Feng, Z. Qian, and N. Dai, "Reversible watermarking via extreme learning machine prediction," *Neurocomputing*, vol. 82, pp. 62–68, 2012.
- [23] Y. Yu, T.-M. Choi, and C.-L. Hui, "An intelligent quick prediction algorithm with applications in industrial control and loading problems," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 276–287, 2012.
- [24] Y.-M. Yang, Y.-N. Wang, and X.-F. Yuan, "Bidirectional extreme learning machine for regression problem and its learning effectiveness," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1498–1505, 2012.
- [25] J. W. Cao and L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *BioMed Research International*, vol. 2014, Article ID 103054, 12 pages, 2014.
- [26] L.-L. Kasun, H. Zhou, G.-B. Huang, and C.-M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.
- [27] G.-B. Huang, H.-M. Zhou, X.-J. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.
- [28] J. Kamran, G. Rafael, and Z. Noureddine, "SW-ELM: a summation wavelet extreme learning machine algorithm with a priori parameter initialization," *Neurocomputing*, vol. 123, pp. 299–307, 2014.
- [29] J.-H. Zhai, H.-Y. Xu, and X.-Z. Wang, "Dynamic ensemble extreme learning machine based on sample entropy," *Soft Computing*, vol. 16, no. 9, pp. 1493–1502, 2012.
- [30] W. He, E.-J. Wang, and T. Xiong, "Intelligent face recognition based on manifold learning and genetic-chaos algorithm optimized Kernel extreme learning machine," *Journal of Communications*, vol. 8, no. 10, pp. 658–664, 2013.
- [31] G. B. Huang, H. M. Zhou, X.-J. Ding, and R. Zhang, "Extreme learning machines for regression and multiclass classification," *IEEE Transaction on Systems Man and Cybernetics B*, vol. 42, no. 2, pp. 513–529, 2011.
- [32] X. Dong, P. Xiaoli, and M. Zhizhong, "Quality prediction of tube hollow based on step-by-step staged MICR," *Chinese Journal of Scientific Instrument*, vol. 28, no. 12, pp. 2190–2196, 2007 (Chinese).

- [33] C. Wang and Y.-H. Gao, "Determination of power distribution network configuration using non-revisiting genetic algorithm," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 3638–3648, 2013.
- [34] M. Murali, M. S. Kumari, and M. Sydulu, "Estimation of locational marginal price in a restructured electricity market with different loss cases using seed genetic algorithm," *Arabian Journal for Science and Engineering*, vol. 39, no. 2, pp. 1089–1106, 2014.