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; zxyang@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 Φ 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 Φ 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 ε_t is smaller than that in previous iteration, ε_{t-1} , the value of Φ 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 Φ 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_{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 trend 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]

$$\text{Minimize: } L_{P_{\text{ELM}}} = \frac{1}{2} \|\beta\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 \quad (1)$$

$$\text{Subject to: } \mathbf{h}(\mathbf{x}_i) \beta = \mathbf{t}_i^T - \xi_i^T, \quad i = 1, \dots, N,$$

where $\xi_i = [\xi_{i,1}, \dots, \xi_{i,m}]^T$ 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:

$$L_{D_{\text{ELM}}} = \frac{1}{2} \|\beta\|^2 + C \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2 - \sum_{i=1}^N \sum_{j=1}^m \alpha_{i,j} (\mathbf{h}(\mathbf{x}_i) \beta_j - t_{i,j} + \xi_{i,j}), \quad (2)$$

where β_j is the vector of the weights between the hidden layer and the j th output node and $\beta = [\beta_1, \dots, \beta_m]$. 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

$$\beta = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (3)$$

The ELM output function is

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \beta = \mathbf{h}(\mathbf{x}) \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}. \quad (4)$$

A kernel matrix for ELM is defined as follows:

$$\Omega_{\text{ELM}} = \mathbf{H}\mathbf{H}^T : \Omega_{\text{ELM},i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \quad (5)$$

Then, the ELM output function can be as follows:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \beta = \mathbf{h}(\mathbf{x}) \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{\text{ELM}}^T \right)^{-1} \mathbf{T}. \quad (6)$$

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 Φ ($0 < \Phi < 1$) 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 Φ , 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 Φ , which is a main factor sensitively affecting the performance of committee machines. If Φ 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 Φ 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 Φ 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 (σ_t stands for the standard deviation, the relative factor λ 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 $\sum_{t=1}^T \alpha_t = 1$.

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 μ_t stands for the expected value, σ_t stands for the standard deviation, and λ 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 $\mu_t, \sigma_t \in (0 - \eta, 0 + \eta)$, $\lim \eta \rightarrow 0$, and $t = 1, \dots, T$, where η 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, σ_t , is solely determined by the individual samples and the generalization performance of the weak learner WL_t . Generally, σ_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 λ is applied on the standard deviation, σ_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 ε_t of each weak learner’s regression model, each weak learner WL_t will be assigned with one accordingly computed weight α_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*

$$E_{\text{ensemble}} \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t (1 - \varepsilon_t)}. \quad (12)$$

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

$$\begin{aligned} \sum_{i=1}^m w_i^{T+1} &\geq \sum_{i \notin I_{T+1}^i} w_i^{T+1} \geq \left(\sum_{i \notin I_{T+1}^i} D(i) \right) \left(\prod_{t=1}^T \beta_t \right)^{1/2} \\ &= E_{\text{ensemble}} \left(\prod_{t=1}^T \beta_t \right)^{1/2}, \end{aligned} \quad (15)$$

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. β_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 β_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_{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_{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, ELM_1 is supplied with m training samples with the uniform distribution of weights in order that each sample owns equal opportunity to be chosen during the first training process for 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, ELM_t , for the input data samples could be represented as one statistics distribution, $\mu_t \pm \lambda\sigma_t$, where μ_t stands for the expected value, and $\lambda\sigma_t$ is defined as robust threshold (σ_t stands for the standard deviation,

and the relative factor λ 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 ELM_t and vice versa for “rejected” predictions. The probabilities of the “rejected” predictions are accumulated to calculate the error rate ε_t . ELM attempts to achieve the $f_t(x)$ with small error rate.

The robust AdaBoost.RT algorithm will calculate the distribution for next ELM_{t+1} . For every sample that is correctly predicted by the current ELM_t , the corresponding weight vector will be multiplied by the error rate function β_t . Otherwise, the weight vector remains unchanged. Such process will be iterated for the next ELM_{t+1} till the last learner ELM_T , unless ε_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 β_t is used as an indicator of regression effectiveness of the ELM_t in the current iteration. According to the relationship between ε_t and β_t , if ε_t increases, β_t will become larger as well. The RAE-ELM will grant a small ensemble weight for the ELM_t . On the other hand, the 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 ε_t in (12) as $\varepsilon_t = 0.5 - \gamma_t$, 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 γ_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 λ in RAE-ELM is the parameter which needs to be optimized within the range $\lambda \in (0, 1)$. We start simulations for λ at 0.1 and increase them at the interval of 0.1. Table 2 shows the examples of setting both T and λ 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 λ and tends to be smaller when λ 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, γ) . Hence, the cost parameter C and the kernel parameter γ 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 γ , that is, 2500 pairs of (C, γ) , are applied as the adjustment parameters. The different values of C and γ 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 λ for Parkinson disease dataset.

λ	T					
	5	10	15	20	25	30
0.1	0.2472	0.2374	0.2309	0.2259	0.2251	0.2249
0.2	0.2469	0.2315	0.2276	0.2241	0.2249	0.2245
0.3	0.2447	0.2298	0.2248	0.2235	0.2230	0.2233
0.4	0.2426	0.2287	0.2249	0.2227	0.2224	0.2227
0.5	0.2428	0.2296	0.2251	0.2219	0.2216	0.2215
0.6	0.2427	0.2298	0.2246	0.2221	0.2214	0.2213
0.7	0.2422	0.2285	0.2257	0.2229	0.2217	0.2219
0.8	0.2441	0.2301	0.2265	0.2228	0.2225	0.2228
0.9	0.2461	0.2359	0.2312	0.2243	0.2237	0.2230

performed combinations of (C, γ) 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 Φ should be chosen before seeing the datasets. In the original AdaBoost.RT-based ensemble ELM, the threshold Φ is required to be manually selected according to an empirical suggestion, which is a sensitive factor affecting the regression performance. If Φ is too low, then it is generally difficult to obtain a sufficient number of “accepted” samples. However if Φ 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 Φ 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 Φ_0 to calculate the followed thresholds in the iterations. Tian and Mao suggested setting the default initial value of Φ_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 Φ_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 λ in RAE-ELM for Parkinson disease dataset.

TABLE 3: Parameters of RAE-ELM and other learning algorithms.

Datasets	RAE ELM (C, L)	Basic ELM [4] (C, L)	Original AdaELM [30] (C, L, ϕ)	Modified AdaELM [31] (C, L, ϕ_0)	SVR [36] (C, γ)	LS SVR [37] (C, γ)
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 Φ . 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 predication 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	0.2571	0.2854	0.2702	0.2653	0.2849	0.2801
Yeast	0.1071	0.1224	0.1195	0.1156	0.1238	0.1182
Computer hardware	0.0563	0.0839	0.0821	0.0726	0.0976	0.0771
Abalone	0.0731	0.0882	0.0793	0.0778	0.0890	0.0815
Servo	0.0727	0.0924	0.0884	0.0839	0.0966	0.0870
Parkinson disease	0.2219	0.2549	0.2513	0.2383	0.2547	0.2437
Housing	0.1018	0.1259	0.1163	0.1135	0.127	0.1185
Cloud	0.2897	0.3269	0.3118	0.3006	0.3316	0.3177
Auto price	0.0809	0.1036	0.0910	0.0894	0.1045	0.0973
Breast cancer	0.2463	0.2641	0.2601	0.2519	0.2657	0.2597
Balloon	0.0570	0.0639	0.0611	0.0592	0.0672	0.0618
Auto-MPG	0.0724	0.0862	0.0799	0.0781	0.0891	0.0857
Bank	0.0415	0.0551	0.0496	0.0453	0.0537	0.0498
Census (house8L)	0.0746	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 predication 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_{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.