Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 946292, 9 pages
http://dx.doi.org/10.1155/2015/946292

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@dlmu.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) ulteriorly 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

$$\sum_{j=1}^{N} \boldsymbol{\beta}_i g\left(\mathbf{w}_j \cdot \mathbf{x}_i + b_j\right) F\left(S\left(\mathbf{x}_i, \mathbf{d}_j\right)\right), \quad i = 1, \ldots, M, \tag{1}$$

where $g(\cdot)$ denotes the activation function. $\mathbf{w}_j$, b_j, and $\boldsymbol{\beta}_j$ are the network weights. $F(\cdot)$ is a fuzzy membership function. $S(\mathbf{x}_i, \mathbf{d}_j)$ is the similarity between the ith input and the jth hidden node. $\mathbf{d}_j \in \mathbb{R}^p$ is the address of the jth 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) $F(x) \to 0, x \to +\infty$.

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,

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{2}$$

the hidden-layer output matrix in LC-ELM is

$$\mathbf{H} = \left[h_{ij} = g\left(\mathbf{w}_j \cdot \mathbf{x}_i + b_j\right) F\left(S\left(\mathbf{x}_i, \mathbf{d}_j\right)\right)\right]_{M \times N}, \quad i = 1, \ldots, M, \quad j = 1, \ldots, N. \tag{3}$$

$\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N]^T_{N \times q}$ is the matrix of output weights and $\boldsymbol{\beta}_i$ denotes the weight vector connecting the ith hidden node and the output layer. $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_M]^T_{M \times q}$ 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 $\{\boldsymbol{\theta}_{k,G} \mid k = 1, 2, \ldots, NP\}$ 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 $\boldsymbol{\theta}_{k,G}$, $k = 1, 2, \ldots, NP$, a mutant vector is generated according to

$$\boldsymbol{\nu}_{k,G+1} = \boldsymbol{\theta}_{r_1,G} + F \cdot \left(\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G}\right) \tag{4}$$

with random and mutually different indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$ 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 $\{(\mathbf{x}_i, \mathbf{t}_i)\, \mathbf{x}_i \in \mathbb{R}^p, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \ldots, M\}$,
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

$$\boldsymbol{\mu}_{k,G+1} = \left(\boldsymbol{\mu}_{1k,G+1}, \boldsymbol{\mu}_{2k,G+1}, \ldots, \boldsymbol{\mu}_{Dk,G+1}\right) \tag{5}$$

is formed such that

$$\boldsymbol{\mu}_{lk,G+1} = \begin{cases} \boldsymbol{\nu}_{lk,G+1} & \text{if rand}\, b\,(l) \le CR \text{ or } l = rnbr\,(k) \\ \boldsymbol{\theta}_{lk,G} & \text{if rand}\, b\,(l) \le CR \text{ or } l \ne rnbr\,(k)\,, \end{cases} \tag{6}$$

where rand $b(l)$ is the lth 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 $rnbr(k)$ is a random chosen integer index $\in$ which ensures that $\boldsymbol{\nu}_{k,G+1}$ 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

$$\boldsymbol{\theta} = \{\mathbf{d}, \mathbf{r}\}\,, \tag{7}$$

where $\mathbf{d} = \{\mathbf{d}_i \mid \mathbf{d}_i \in \mathbb{R}^p, i = 1, \ldots, N\}$ 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

$$E = \sqrt{\frac{\sum_{i=1}^{M} \left\| \sum_{j=1}^{N} \boldsymbol{\beta}_i h_{ij} - \mathbf{t}_i \right\|_2^2}{M \times q}}. \tag{8}$$

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 $\{(\mathbf{x}_i, \mathbf{t}_i)\, \mathbf{x}_i \in \mathbb{R}^p, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \ldots, M\}$;
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 $\mathbf{r}$.
(2) $Iter = 1$.
(3) **while** $Iter \leqslant Itermax$ **do**
(4) (1) Calculate the hidden layer output matrix $\mathbf{H}$,
(2) Calculate the output weight $\boldsymbol{\beta}$,
(3) Adjust ($\mathbf{d}$, $\mathbf{r}$) 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:

$$\boldsymbol{\theta}_{k,G+1} = \begin{cases} \boldsymbol{\mu}_{k,G} & \text{if } f\left(\boldsymbol{\theta}_{k,G}\right) - f\left(\boldsymbol{\mu}_{k,G}\right) > \varepsilon f\left(\boldsymbol{\theta}_{k,G}\right), \\ \boldsymbol{\mu}_{k,G} & \text{if } \left|f\left(\boldsymbol{\theta}_{k,G}\right) - f\left(\boldsymbol{\mu}_{k,G}\right)\right| < \varepsilon f\left(\boldsymbol{\theta}_{k,G}\right), \\ & \left\|\boldsymbol{\beta}^{\boldsymbol{\mu}_{k,G}}\right\| < \left\|\boldsymbol{\beta}^{\boldsymbol{\theta}_{k,G}}\right\|, \\ \boldsymbol{\theta}_{k,G} & \text{else}, \end{cases} \tag{9}$$

where $f(\cdot)$ is the fitness function (RMSE) and ε 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:

$$F(x) = \exp\left(-\frac{x^2}{r}\right), \tag{10}$$

$$F(x) = \frac{2}{1 + \exp(x/r)}, \tag{11}$$

$$F(x) = \tanh\left(-\frac{x}{r}\right) + 1. \tag{12}$$

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

$$G(x, y) = \frac{1}{2\pi \times 0.5^2} \exp\left(-\frac{x^2 + y^2}{2 \times 0.5^2}\right) \cos\left(2\pi\left(x + y\right)\right). \tag{13}$$

Table 1: Configurations of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM.

Configuration	ELM	OSFuzzyELM	LC-ELM	ELC-ELM
Input weights && 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 && window radius	N/A	N/A	RN in $[0, 1]$ && 0.4	RN in $[0, 1]$
Similarity	N/A	N/A	Wave kernel	Wave kernel
Fuzzy membership function	N/A	(10)	(11)	(11)

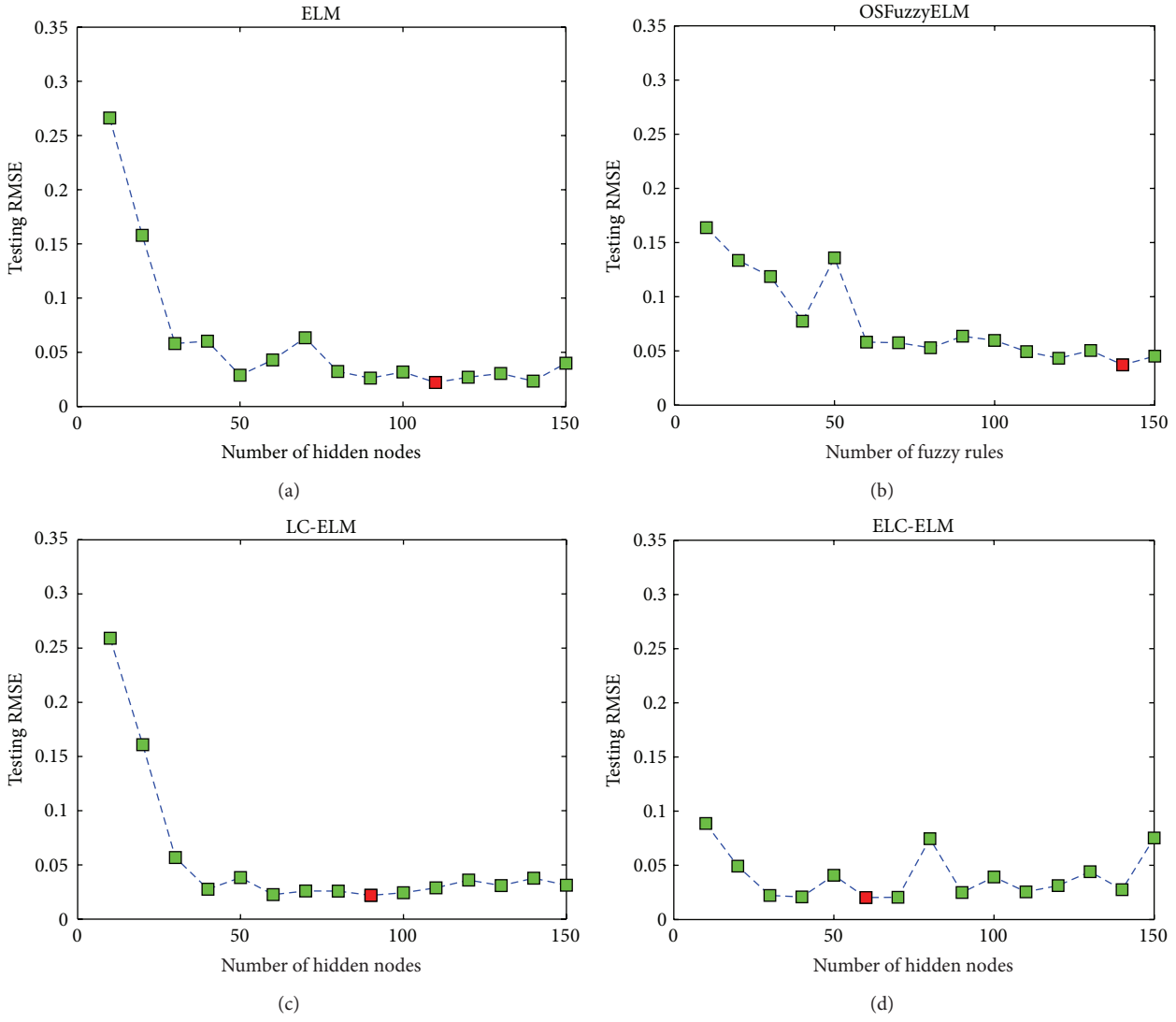


Figure 1: Testing RMSEs of ELM, OSFuzzyELM, LC-ELM, and ELC-ELM with respect to different numbers of hidden nodes.

In this example, 51×51 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 RMSE	Hidden nodes	Avg. training		Avg. testing	
			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 of ELM, OSFuzzyELM, and LC-ELM on several benchmark datasets [18]. The specifications of the datasets are displayed in Table 5.

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

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	**0.0401**	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	**0.0769**	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	**0.1129**	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	5.6467×10^3	**0.0288**	0.0005	1.0868	125
CPU	ELM	0.0476	0.0066	0.0016	0.0865	0.0584	0.0008	10
	OSFuzzyELM	**0.0284**	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	**0.0360**	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	**0.0807**	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
Ecoli	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	**90.89**	0.74	312.5917	**89.29**	4.29	0.0162	20
Glass	ELM	72.25	3.82	0.0047	63.47	5.36	0.0016	20
	OSFuzzyELM	**92.04**	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	**66.39**	6.98	0.0074	20
Ionosphere	ELM	91.70	1.69	0.0078	83.51	3.12	0.0031	40
	OSFuzzyELM	**95.88**	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	**97.20**	2.70	0.0054	15
Sonar	ELM	88.70	2.92	0.0031	75.57	3.95	0.0016	40
	OSFuzzyELM	**97.46**	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	**78.00**	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	**97.54**	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.

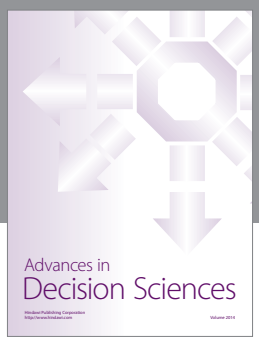
[17] R. Jensen and Q. Shen, "New approaches to fuzzy-rough feature selection," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 824–838, 2009.

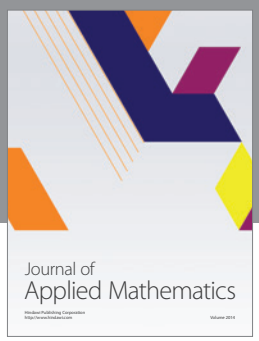
[18] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

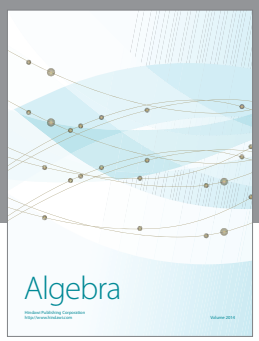
[19] M. Mike, *Statistical Datasets*, Department of Statistics, Carnegie Mellon University, 1989.

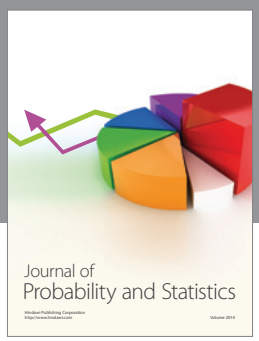
[20] Y. Qu, C. Shang, Q. Shen, N. M. Parthaláin, and W. Wu, "Kernel-based fuzzy-rough nearest neighbour classification," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ '11)*, pp. 1523–1529, June 2011.

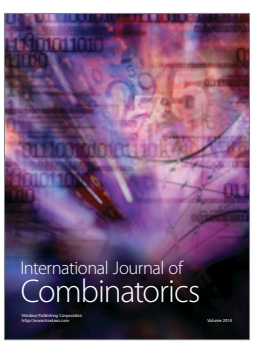
[21] C. Shang, D. Barnes, and Q. Shen, "Facilitating efficient mars terrain image classification with fuzzy-rough feature selection," *International Journal of Hybrid Intelligent Systems*, vol. 8, no. 1, pp. 3–13, 2011.

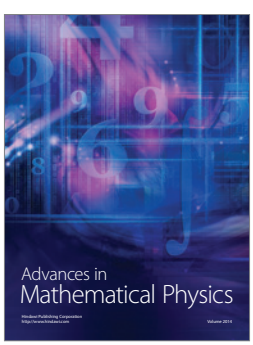


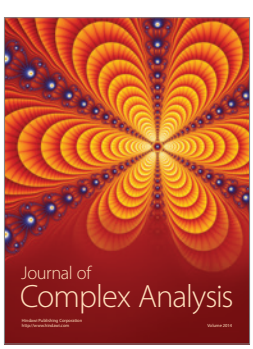


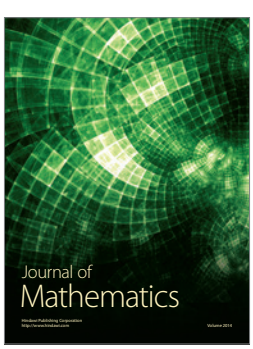


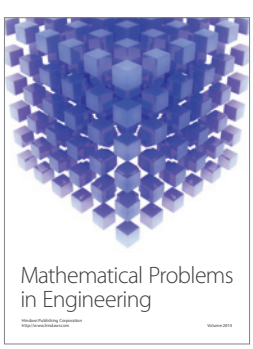


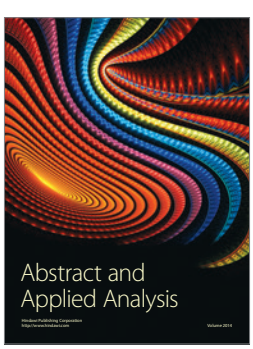


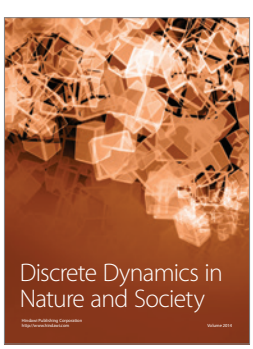


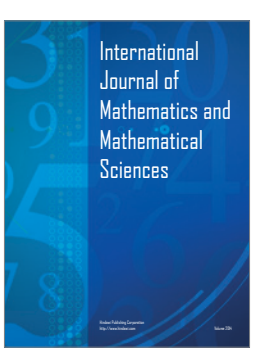












Hindawi

Submit your manuscripts at
http://www.hindawi.com