Neurocomputing 74 (2011) 2483–2490

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A study on effectiveness of extreme learning machine[☆]

Yuguang Wang, Feilong Cao[*], Yubo Yuan

Department of Mathematics, China Jiliang University, Hangzhou 310018, Zhejiang Province, PR China

ARTICLE INFO

Available online 12 May 2011

Keywords:
Feedforward neural networks
Extreme learning machine
Effective extreme learning machine

ABSTRACT

Extreme learning machine (ELM), proposed by Huang et al., has been shown a promising learning algorithm for single-hidden layer feedforward neural networks (SLFNs). Nevertheless, because of the random choice of input weights and biases, the ELM algorithm sometimes makes the hidden layer output matrix **H** of SLFN not full column rank, which lowers the effectiveness of ELM. This paper discusses the effectiveness of ELM and proposes an improved algorithm called EELM that makes a proper selection of the input weights and bias before calculating the output weights, which ensures the full column rank of **H** in theory. This improves to some extend the learning rate (testing accuracy, prediction accuracy, learning time) and the robustness property of the networks. The experimental results based on both the benchmark function approximation and real-world problems including classification and regression applications show the good performances of EELM.

1. Introduction

Extreme learning machine (ELM) proposed by Huang et al. shows as a useful learning method to train single-hidden layer feedforward neural networks (SLFNs) which have been extensively used in many fields because of its capability of directly approximating nonlinear mappings by input data and providing models for a number of natural and artificial problems that are difficult to cope with by classical parametric techniques. So far there have been many papers addressing relative problems. We refer the reader to [8,12,14–18,20,24,27,28].

In theory, many researchers have explored the approximation ability of SLFNs. In 1989, Cybenko [7] and Funahashi [10] proved that any continuous functions can be approximated on a compact set with uniform topology by an SLFN with any continuous, sigmoidal activation function, which made a breakthrough in the artificial neural network field. Leshno [19] improved the results of Hornik [11] and proved that any continuous functions could be approximated by feedforward networks with a nonpolynomial activation function. Furthermore, some deep and systematic studies on the condition of activation function can be found in [4–6]. Recently, Cao et al. [3] constructively gave the estimation of upper bounds of approximation for continuous functions by SLFNs with the bounded, strictly monotone and odd activation function, which means that the neural networks can be constructed without training as long as the samples are given. In practical applications, for function approximation in a finite training set, Huang and Babri [13] showed that an SLFN with at most N hidden nodes and with almost any nonlinear activation function can exactly learn N distinct samples.

In comparison with other traditional learning methods such as BP algorithm, the ELM algorithm proves a faster learning algorithm for SLFNs. There are some advantages of the ELM algorithm: (1) easy to use and no parameters need to be tuned except predefined network architecture; (2) it is proved a faster learning algorithm compared to other conventional learning algorithms such as BP algorithm. Most training can be accomplished in seconds and minutes (for large-scale complex applications) which might not be easily obtained using other traditional learning methods; (3) it possesses similar high generalization performance as BP and SVM; (4) a wide range of activation functions including all piecewise continuous functions can be used as activation functions in ELM. Among the above four features, the most impressive one is the fast speed of training which is far superior to other conventional learning methods.

However, there are also some shortcomings that the ELM algorithm cannot overcome. Random choosing of input weights and biases easily causes the so-called hidden layer output matrix not full column rank. This sometimes makes the linear system that is used to train output weights (linking the hidden layer to the output layer) unsolvable. It also lowers the predicting accuracy. Bartlett [1] pointed out that for feedforward neural networks the smaller the norm of weights and training error are, the better generalization performance the networks tend to have. Therefore, it is necessary to develop a more effective learning

[☆]The research was supported by the National Natural Science Foundation of China (Nos. 60873206, 61001200), the Foundation of Innovation Team of Science and Technology of Zhejiang Province of China (No. 2009R50024) and the Innovation Foundation of Post-Graduates of Zhejiang Province of China (No. YK2008066).

[*] Corresponding author.
E-mail address: feilongcao@gmail.com (F. Cao).

0925-2312/$ - see front matter Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.11.030

method that can overcome this shortcoming and approximate as fast as the ELM algorithm.

This paper tries to design such a learning machine. To achieve this, we first properly train the input weights and biases such that the hidden layer output matrix is full column rank. Then, a new learning algorithm called effective extreme learning machine (EELM) is proposed, where the strictly diagonally dominant criterion for determining a nonsingular matrix is used to choose the proper input weights and biases. In the first phase of the EELM algorithm the samples are sorted by affine transformation. Due to the assumption of the constructive algorithm, the activation function of the network used in our algorithm should be a Gaussian radial basis-type function. With sorted samples, the Gaussian radial basis-type activation function helps distinguish diagonal elements of the matrix from other non-diagonal ones such that the diagonal elements is larger than the sum of all absolutes of non-diagonal elements. Having chosen input weights and biases properly such that the hidden layer output matrix is full column rank, simple generalized inverse operation gives the output weights.

The difference between the new proposed EELM algorithm and the ELM algorithm lies in the training of input weights and biases. And time spent in the first phase of EELM is very short compared to the second step. So EELM is actually faster than ELM. And EELM algorithm also possesses the qualities of the ELM algorithm including easy implementing, good generalization performance. Moreover, the new algorithm improved the effectiveness of learning: the full column rank property of the matrix makes the orthogonal projection method, a fast algorithm for solving generalized inverse, available. So, it is called effective extreme learning machine.

This paper is organized as follows. Section 2 gives two theorems that show the two steps in the first phase (training input weights and biases) of EELM algorithm are strictly correct and reasonable theoretically. The constructive proofs in the theorems actually provide the learning method. Section 3 proposes the new EELM learning algorithm for SLFNs. In Section 4, the complexity of the algorithm is given and performance is measured. Section 5 consists of the discussions and conclusions.

2. Proper choice of input weights and biases to improve the effectiveness of learning

For n arbitrary distinct samples $\{(X_i,t_i)|i=1,2,\ldots,n\}$ where $X_i=(x_{i1},x_{i2},\ldots,x_{id})^T \in \mathbb{R}^d$ and $t_i=(t_{i1},t_{i2},\ldots,t_{im})\in\mathbb{R}^m$, standard SLFNs with N nodes and activation function g are mathematically modeled as

$$G_N(X)=\sum_{i=1}^{N}\beta_i g(W_i\cdot X+b_i),$$

here $\beta_i=(\beta_{i1},\beta_{i2},\ldots,\beta_{iN})^T$ is the output weight vector connecting the i-th nodes and output nodes, $W_i=(w_{i1},w_{i2},\ldots,w_{id})^T$ is the input weight vector connecting the i-th hidden nodes and the input nodes, and the b_i is the threshold of the i-th hidden node. Approximating the samples with zero error means the proper selection of β_i, W_i and b_i such that

$$\|G_N(X_j)-t_j\|=0 \quad (j=1,2,\ldots,n)$$

or

$$G_N(X_j)=t_j \quad (j=1,2,\ldots,n), \tag{1}$$

that is,

$$\mathbf{H}\beta=T,$$

here

$$\mathbf{H}=\mathbf{H}(w_1,w_2,\ldots,w_N,b_1,b_2,\ldots,b_N,X_1,X_2,\ldots,X_n) = (h_{ij})_{n\times n}=\begin{pmatrix} g(W_1\cdot X_1+b_1) & \cdots & g(W_N\cdot X_1+b_N)\\ \vdots & \cdots & \vdots\\ g(W_1\cdot X_n+b_1) & \cdots & g(W_N\cdot X_n+b_N)\end{pmatrix}. \tag{2}$$

As named in Huang et al. [13,12], **H** is called the hidden layer output matrix of the neural networks.

In ELM algorithm, the choice of β_i and W_i is random, which accelerates the rate of learning. Nevertheless, the random selection sometimes produces nonsingular hidden layer output matrix which causes no solution of the linear system (1). To overcome the shortcoming, an extra phase of training the input weights and biases within acceptable steps to keep **H** full column rank should be added to the algorithm.

First, we introduce the definition of inverse lexicographical order (or inverse dictionary order) in $\mathbb{R}^d$.

Definition 2.1. Suppose $X_1,X_2\in\mathbb{R}^d$ where $X_i=(x_{i1},x_{i2},\ldots,x_{id})\in\mathbb{R}^d$ $(i=1,2)$ are defined as $X_1<_d X_2$ if and only if there exists $j_0\in\{1,2,\ldots,d\}$ such that $x_{1j_0}<x_{2j_0}$ and $x_{1j}=x_{2j}$ for $j=j_0+1,\ldots,d$. j_0 is called different attribute and denoted by $da(X_1,X_2)$ or $da(1,2)$ for convenience if no confusion is produced.

With the concept of inverse lexicographical order, we obtain the following theorem that gives a constructive method for sorting high-dimensional vectors via an affine transformation.

Theorem 2.2. *For n distinct vectors $X_1<_d X_2<_d\cdots<_d X_n\in\mathbb{R}^d$ $(d\ge 2)$ and $X_i=(x_{i1},x_{i2},\ldots,x_{id})^T$ such that $\sum_{j=1}^d x_{ij}^2>0$ for each $i=1,2,\ldots,n$. Calculate $W\in\mathbb{R}^d$ as follows*:

$$w_j^1=\frac{1}{\max_{i=1,2,\ldots,n}\{|x_{ij}|\}}>0 \quad (j=1,2,\ldots,d),$$

$$x_{ij}^1=w_j^1 x_{ij}\in[-1,1] \quad (i=1,2,\ldots,n,\ j=1,2,\ldots,d),$$

$$y_{ij}^1=|x_{i+1,j}^1-x_{ij}^1| \quad (i=1,2,\ldots,n-1,\ j=1,2,\ldots,d),$$

$$\delta=\log_{10}d+\log_{10}2,$$

$$n_j=\left[-\log_{10}\left(\min_{i=1,2,\ldots,n}\{y_{ij}^1\}\right)\right]+\delta \quad (j=1,2,\ldots,d),$$

$$w_j^2=w_j^1 10^{\sum_{p=1}^{j} n_p} \quad (j=1,2,\ldots,d),$$

$$W=(w_1^2,w_2^2,\ldots,w_d^2).$$

Then follows:

$$W\cdot X_1<W\cdot X_2<\cdots<W\cdot X_n.$$

Proof. For each fixed $i=1,2,\ldots,n-1$, set $k_0=da(i,i+1)$ which means by Definition 2.1 that $x_{ik_0}<x_{i+1,k_0}$ and $x_{ij}=x_{i+1,j}$ $(j=k_0+1,\ldots,d)$. Then

$$W\cdot X_{i+1}-W\cdot X_i =\sum_{k=1}^{k_0-1}(x_{i+1,k}^1-x_{ik}^1)10^{\sum_{p=1}^{k} n_p}+(x_{i+1,k_0}^1-x_{ik_0}^1)10^{\sum_{p=1}^{k_0} n_p},$$

where by definition $x_{i,k}^1,x_{i+1,k}^1\in[-1,1]$ $(k=1,2,\ldots,d)$ and $x_{i,k_0}^1<x_{i+1,k_0}^1$. Noticing

$$10^{\sum_{p=1}^{k} n_p}=10^{\sum_{p=1}^{k_0-1} n_p}\left(\prod_{p=k+1}^{k_0-1}10^{n_p}\right)^{-1}\le\frac{10^{\sum_{p=1}^{k_0-1} n_p}}{d^{(k_0-1)-k}} \quad (k=1,2,\ldots,k_0-1)$$

and

$$10^{n_{k_0}} \geq 10^{-\log_{10}(\min_{i=1,2,\ldots,n}\{y^1_{ik_0}\})+\delta} = \frac{2d}{\min_{i=1,2,\ldots,n}\{y^1_{ik_0}\}}.$$

Therefore,

$$\begin{aligned} W\cdot X_{i+1} - W\cdot X_i &\geq -2\sum_{k=1}^{k_0-1} 10^{\sum_{p=1}^{k} n_p} + (y^1_{ik_0}10^{n_{k_0}})10^{\sum_{p=1}^{k_0-1} n_p} \\ &\geq -(2d)\sum_{k=1}^{k_0-1} d^{-k}10^{\sum_{p=1}^{k_0-1} n_p} + (2d)10^{\sum_{p=1}^{k_0-1} n_p} \\ &= (2d)10^{\sum_{p=1}^{k_0-1} n_p}\left(1-\frac{1-d^{-(k_0-1)}}{d-1}\right) > 0. \end{aligned}$$

This completes the proof of Theorem 2.2. □

Remark 1. For the case of $d=1$, one need not follow the steps of Theorem 2.2 when selecting W. In fact, in the case of $d=1$, the sort operation of the samples $X_1, X_2, \ldots, X_n$ in the inverse lexicographical order via affine transformation can be skipped over. In addition, we do not need the prior sorting of the samples $X_1, X_2, \ldots, X_n$ in the inverse lexicographical order. This is because the computation of W is independent of the order of the samples. Therefore, one only has to sort $W\cdot X_1, W\cdot X_2, \ldots, W\cdot X_n$ and change the order of $t_1, t_2, \ldots, t_n$ correspondingly in the linear system.

Having calculated the W and given an order to $W\cdot X_1, W\cdot X_2, \ldots, W\cdot X_n$, we are able to select input weights and biases, which ensures nonsingularity of the hidden layer output matrix of the neural networks. This is stated in the following theorem. It is noteworthy that the activation function should satisfy some assumptions.

Theorem 2.3. *Let $g(x)$ be a positive finite function on $\mathbb{R}$ such that $\lim_{x\to-\infty} g(x) = \lim_{x\to+\infty} g(x) = 0$ and $g(x)$ is not identically equal to 0. Given n distinct samples $X_1 <_d X_2 < \cdots <_d X_n \in \mathbb{R}^d$ $(d \geq 2)$ and $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$ such that $\sum_{j=1}^{d} x_{ij}^2 > 0$ for any $i=1,2,\ldots,n$, there exist input weights $W_i \in \mathbb{R}^d$ and biases $b_i \in \mathbb{R}$ $(i=1,2,\ldots,n)$ such that the square matrix*

$$\mathbf{H} = (h_{ij})_{n\times n} = \begin{pmatrix} g(W_1\cdot X_1+b_1) & g(W_2\cdot X_1+b_2) & \cdots & g(W_n\cdot X_1+b_n) \\ g(W_1\cdot X_2+b_1) & g(W_2\cdot X_2+b_2) & \cdots & g(W_n\cdot X_2+b_n) \\ \vdots & \vdots & \cdots & \vdots \\ g(W_1\cdot X_n+b_1) & g(W_2\cdot X_n+b_2) & \cdots & g(W_n\cdot X_n+b_n) \end{pmatrix}$$

is nonsingular.

Proof. The proof of the theorem is actually the process of selection of input weights and biases. By assumptions that $g(x)$ is a finite function on $\mathbb{R}$ and $\lim_{x\to-\infty} g(x) = \lim_{x\to+\infty} g(x) = 0$, $g(x)$ has a maximum on $\mathbb{R}$. Set $M = g(x_0) = \max\{g(x)|x\in\mathbb{R}\}$ $(x_0 \in \mathbb{R})$. For $\varepsilon_0 = M/n > 0$, there exists $a > 0$ such that $g(x) < M/n^2$ for $|x| > a$ and $x_0 \in (-a,a)$. Now choose $W=1$ if $d=1$ and W by Theorem 2.2 if $d \geq 2$, which implies by assumptions and Theorem 2.2 that

$$W\cdot X_1 < W\cdot X_2 < \cdots < W\cdot X_n. \tag{3}$$

Then, select W_i and b_i $(i=1,2,\ldots,n)$ as follows:

$$dist = \max\{a-x_0, a+x_0\}, \tag{4}$$

$$k_i = \frac{2dist}{\min\{W\cdot X_{i+1}-W\cdot X_i, W\cdot X_i - W\cdot X_{i-1}\}} \quad (i=2,3,\ldots,n-1), \tag{5}$$

$$W_i = k_i W \ (i=2,3,\ldots,n-1), \quad W_1 = W_2, \quad W_n = W_{n-1},$$

$$b_i = x_0 - W_i\cdot X_i \quad (i=1,2,\ldots,n). \tag{6}$$

One thus obtains by (6) that for $i=1,2,\ldots,n$,

$$g(W_i\cdot X_i + b_i) = M.$$

By (3)–(5), there holds for $i=1,2,\ldots,n$,

$$|(W_i\cdot X_j+b_i)-(W_i\cdot X_i+b_i)| \geq k_i|W\cdot X_j - W\cdot X_i| \geq 2dist \quad (j=1,2,\ldots,n,\ j\neq i)$$

and

$$W_i\cdot X_1+b_i < W_i\cdot X_2+b_i < \cdots W_i\cdot X_{i-1}+b_i < x_0-a < W_i\cdot X_i+b_i = x_0 < x_0+a < W_i\cdot X_{i+1}+b_i < \cdots < W_i\cdot X_n+b_i.$$

Therefore,

$$g(W_i\cdot X_j+b_i) < \frac{M}{n^2}.$$

Hence,

$$h_{ii} > \sum_{\substack{(k,i)\neq(i,i)\\ k,j=1,2,\ldots,n}} |h_{kj}|,$$

thus, **H** is strictly diagonally dominant. So **H** is nonsingular. This completes the proof of Theorem 2.3. □

Remark 2. We summarize the steps of selection of input weights and biases as follows. First, choose $a \in \mathbb{R}$ such that

$$g(x) < \frac{M}{n^2} \quad (|x| > a,\ x_0 \in (-a,a)),$$

here,

$$M = \max\{g(x)|x\in\mathbb{R}\} = g(x_0) \quad (x_0 \in \mathbb{R}).$$

Then, calculate as follows:

$$dist = \max\{a-x_0, a+x_0\},$$

$$k_i = \frac{2dist}{\min\{W\cdot X_{i+1}-W\cdot X_i, W\cdot X_i - W\cdot X_{i-1}\}} \quad (i=2,3,\ldots,n-1),$$

$$W_i = k_i W \ (i=2,3,\ldots,n-1), \quad W_1 = W_2, \quad W_n = W_{n-1},$$

$$b_i = x_0 - W_i\cdot X_i \quad (i=1,2,\ldots,n).$$

Remark 3. Actually, there exist activation functions meeting the conditions of Theorem 2.3. One kind of such activation functions is functions with one peak such as Gaussian radial basis function $g(x) = e^{-x^2}$.

Remark 4. In the case that the number of rows m of **H** is larger than that of columns, W and b are calculated based on the square matrix which consists of m forward rows of **H**.

3. Proposed effective extreme learning machine (EELM)

Based upon Theorems 2.2 and 2.3, a more effective method for training SLFNs is proposed in this section.

3.1. Features of extreme learning machine (ELM) algorithm

The ELM algorithm proposed by Huang et al. can be summarized as follows.

Algorithm ELM. Given a training set $\mathcal{N} = \{(X_i, t_i)| \in \mathbb{R}^d,\ t_i \in \mathbb{R}, i = 1,2,\ldots,n\}$ and activation function g, hidden node number n_0.

Step 1: Randomly assign input weight W_i and bias b_i $(i=1,2,\ldots,n_0)$.
Step 2: Calculate the hidden layer output matrix **H**.

Step 3: Calculate the output weight β by $\beta = \mathbf{H}^{\dagger}T$, here $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of $\mathbf{H}$ (see [21,25] for further details) and $T = (t_1, t_2, \ldots, t_n)^T$.

The ELM is proved in practice as an extremely fast algorithm. This is because it randomly chooses the input weights W_i and biases b_i of the SLFNs instead of selection. However, this big advantage makes the algorithm less effective sometimes. As mentioned in Section 1, the random choice of input weights and biases is likely to produce an unexpected result, that is, the hidden layer output matrix $\mathbf{H}$ is not full column rank or singular (see (2)), which causes two difficulties. First, it enlarges training error of the samples, which to some extent lowers the prediction accuracy as we can see in the following sections. Besides, the ELM cannot use the orthogonal projection method to calculate Moore–Penrose generalized inverse of $\mathbf{H}$ due to the singularity of $\mathbf{H}$, instead, it prefers singular value decomposition (SVD) which wastes more time.

3.2. Improvement for the effectiveness of extreme learning machine

According to Theorems 2.2 and 2.3, one can summarize the new extreme learning machine for SLFNs as follows. We call the new algorithm effective extreme learning machine. In the algorithm, Gaussian radial basis activation function is used.

Algorithm EELM. Given a training data set $\mathcal{N} = \{(X_i^*, t_i^*) | X_i^* \in \mathbb{R}^d, t_i^* \in \mathbb{R}, i = 1, 2, \ldots, n\}$, activation function of radial basis function $g(x) = e^{-x^2}$ and hidden node number n_0.

Step 1: Select weights W_i and bias b_i $(i = 1, 2, \ldots, n_0)$.

Sort $W \cdot X_1^*, W \cdot X_2^*, \ldots, W \cdot X_{n_0}^*$ in order that $W \cdot X_{i_1}^* < W \cdot X_{i_2}^* < \cdots < W \cdot X_{i_{n_0}}^*$ $(i_j \neq i_k$ for $j \neq k$, $j,k = 1, 2, \ldots, n_0$ and $i_j = 1, 2, \ldots, n_0)$ are satisfied, then correspondingly change the order of the forward n_0 samples (X_i^*, t_i^*) $(i = 1, 2, \ldots, n_0)$. And denote the sorted data by (X_i, t_i) $(i = 1, 2, \ldots, n)$ and $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ $(i = 1, 2, \ldots, n)$. For $j = 1, 2, \ldots, d$, make the following calculations:

$$w_j^1 = \frac{1}{\max_{i=1,2,\ldots,n_0}\{|x_{ij}|\}} > 0,$$

$$x_{ij}^1 = w_j^1 x_{ij} \in [-1, 1],$$

$$y_{ij}^1 = |x_{i+1,j}^1 - x_{ij}^1| \quad (i = 1, 2, \ldots, n_0 - 1),$$

$$\delta = \log_{10} d + \log_{10} 2,$$

$$n_j = \left[-\log_{10}\left(\min_{i=1,2,\ldots,n}\{y_{ij}^1\} \right) \right] + \delta,$$

$$w_j^2 = w_j^1 10^{\sum_{p=1}^{j} n_p}.$$

Set

$$W = (w_1^2, w_2^2, \ldots, w_d^2).$$

Let

$$M = 1, \quad x_0 = 0, \quad a = \max\{\sqrt{|2\ln(n_0)|}, 1\} + 1$$

such that $M = \max\{g(x) | x \in \mathbb{R}\} = g(x_0)$ $(x_0 \in \mathbb{R})$ and $g(x) < M / n_0^2$ $(|x| > a$, $x_0 = 0 \in (-a, a))$. Then,

$$dist = \max\{a - x_0, a + x_0\},$$

$$k_i = \frac{2dist}{\min\{W \cdot X_{i+1} - W \cdot X_i, W \cdot X_i - W \cdot X_{i-1}\}} \quad (i = 2, 3, \ldots, n_0 - 1),$$

$$W_i = k_i W \ (i = 2, 3, \ldots, n_0 - 1), \quad W_1 = W_2, \quad W_{n_0} = W_{n_0 - 1},$$

$$b_i = x_0 - W_i \cdot X_i \quad (i = 1, 2, \ldots, n_0).$$

Step 2: Calculate output weights $\beta = (\beta_1, \beta_2, \ldots, \beta_{n_0})$ $(i = 1, 2, \ldots, n_0)$.

Let $T = (t_1, t_2, \ldots, t_n)^T$ and

$$\mathbf{H} = \begin{pmatrix} g(W_1 \cdot X_1 + b_1) & g(W_2 \cdot X_1 + b_2) & \cdots & g(W_{n_0} \cdot X_1 + b_{n_0}) \\ \vdots & \vdots & \cdots & \vdots \\ g(W_1 \cdot X_{n_0} + b_1) & g(W_2 \cdot X_{n_0} + b_2) & \cdots & g(W_{n_0} \cdot X_{n_0} + b_{n_0}) \\ g(W_1 \cdot X_{n_0+1} + b_1) & g(W_2 \cdot X_{n_0+1} + b_2) & \cdots & g(W_{n_0} \cdot X_{n_0+1} + b_{n_0}) \\ \vdots & \vdots & \cdots & \vdots \\ g(W_1 \cdot X_n + b_1) & g(W_2 \cdot X_n + b_2) & \cdots & g(W_{n_0} \cdot X_n + b_{n_0}) \end{pmatrix}_{n \times n_0}.$$

Then

$$\beta = \mathbf{H}^{\dagger} T = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T T.$$

Remark 5. As pointed out in Remark 3, the activation function in the EELM algorithm for SLFNs should be chosen to satisfy the assumption of Theorem 2.3. By Theorem 2.3, if the samples possess the properties that they are distinct and $\sum_{j=1}^{d} x_{ij}^2 > 0$ (this is actually almost surely), then the hidden layer output matrix $\mathbf{H}$ is full column rank. This ensures that $\mathbf{H}^T\mathbf{H}$ is nonsingular and thus the fast orthogonal project method can be used in the computation of Moore–Penrose generalized inverse. Moreover, when $n = n_0$, that is, $\mathbf{H}$ is an invertible square matrix.

Remark 6. In accordance with Theorem 2.3, $X_1 <_d X_2 <_d \cdots <_d X_{n_0}$, here the (X_i, t_i) $(i = 1, 2, \ldots, n)$ denotes the sorted samples as in the algorithm above. To avoid the random error, the n_0 samples being used to train input weights and biases can be randomly chosen from the original n samples.

4. Complexity and performance

The proposed EELM spends more time on training samples than ELM but in fact the extra time spent on selecting input weights and biases is $\mathcal{O}(n_0 d)$. Compared with the second phase of the algorithm that calculates the output weights, it can be viewed as a constant.

In the rest of this section, the performance of the proposed EELM learning algorithm is measured compared with the ELM algorithm. The simulations for ELM and EELM algorithms are carried out in the Matlab 7.0 environment running in Intel Celeron 743 CPU with the speed of 1.30 GHz and in Intel Core 2 Duo CPU. The activation function used in both algorithms is Gaussian radial basis function $g(x) = e^{-x^2}$.

4.1. Benchmarking with a regression problem: approximation of 'SinC' function with noise

First of all, we use the 'SinC' function to measure the performance of the two algorithms. The target function is as follows:

$$y = f(x) = \begin{cases} \sin(x)/x, & x \neq 0, \\ 1, & x = 0. \end{cases}$$

A training set (X_i, t_i) and testing set (X_i, t_i) with 200 samples are respectively created, where X_i in the training data is distributed in $[-10, 10]$ with uniform step length. The X_i in the testing data is chosen randomly in the standard normalized distribution in $[-30, 30]$. The reason why the range $([-30, 30])$ of testing data is longer than that of training data is because an obvious way to assess the quality of the learned model is to see on how long term the predictions given by the model are accurate. The experiment is carried out on these data as follows. There are 200 hidden nodes assigned for both the ELM and the EELM algorithms. Fifty trials have been conducted for the ELM algorithm to eliminate the random error and the results shown are the average results. Results shown in Table 1 include training time, testing time, training accuracy, testing accuracy and the number of nodes of both algorithms.

It can be seen from Table 1 that the EELM algorithm spent 0.0624 *s* CPU time obtaining testing accuracy 0.1595 with zero training error whereas it takes 0.0870 *s* CPU time for the ELM algorithm to reach a higher error 5.6642 with training error of 3.1431×10^{-6}. Fig. 2 shows the expected and approximated results of EELM algorithm and Fig. 1 shows the true and the approximated results of ELM algorithm. The results show that our EELM algorithm can not only approximate the training data with zero error but also have a long term prediction accuracy. And the time consumption is not more than the ELM algorithm. Though the ELM algorithm has good performance in the interval $[-10,10]$, its long term prediction accuracy is not very satisfactory.

4.2. Benchmarking with real-world applications

In this section, we conduct the performance comparison of the proposed EELM and the ELM algorithms for five real problems: three classification tasks including Diabetes, Glass Identification (Glass ID), Statlog (Landsat Satellite), and two regression tasks including Housing and Slump (Concrete Slump). All the data sets are from UCI repository of machine learning databases [2]. The speculation of each database is shown in Table 2. For the databases that have only one data table, as conducted in [9,22,23,26], 75% and 25% of samples in the problem are randomly chosen for training and testing respectively at each trial.

Fifty trials were conducted for the two algorithms and the results are reported in Tables 3–5, which show that in our simulation, generally speaking, ELM can reach higher testing rate for mid and large size classification problems than EELM and for the small size ones, EELM can achieve a higher rate than ELM.

Table 1
Performance comparison for learning function: SinC.

Algorithms	Time		Accuracy		No. of nodes
	Training	Testing	Training	Testing	
ELM	0.0870	0.0056	3.1431×10^{-6}	5.6642	200
EELM	0.0624	0	0.0038	0.1595	200

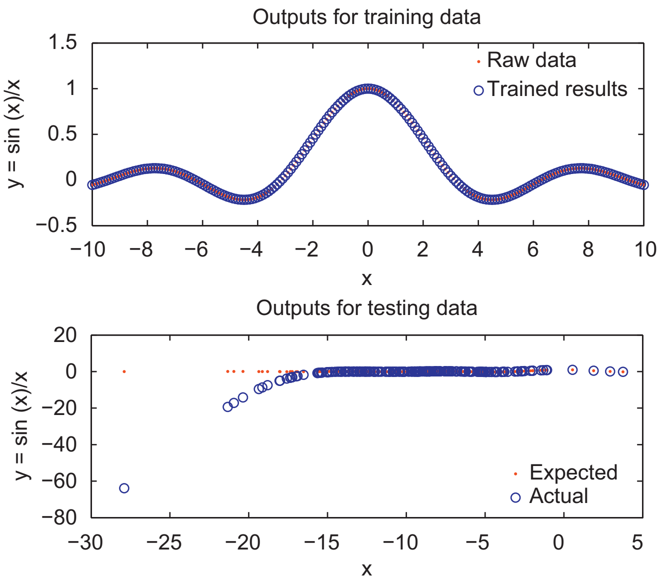

Fig. 1. Outputs of ELM learning algorithm.

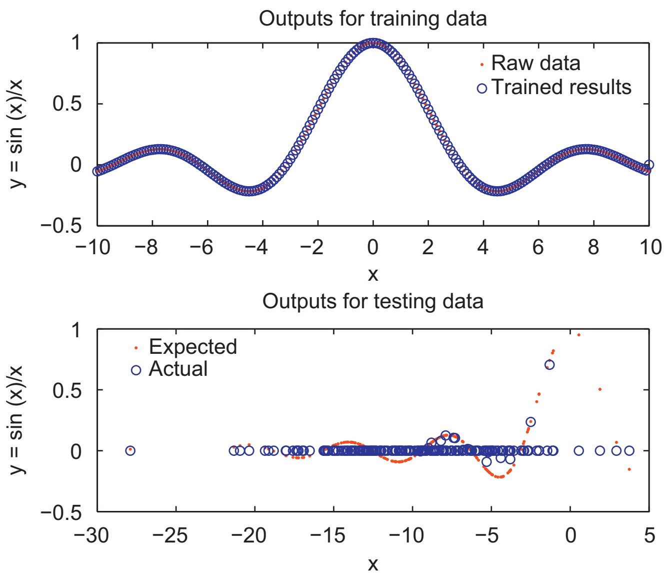

Fig. 2. Outputs of EELM learning algorithm.

Table 2
Speculations of real-world applications and the number of nodes for each.

Data sets	# Observations		# Attributes	Associated tasks	#Nodes
	Training	Testing	Continuous		
Diabetes	576	192	8	Classification	20
Statlog	4435	2000	36	Classification	20
Glass ID	160	54	9	Classification	10
Housing	378	126	14	Regression	80
Slump	76	27	10	Regression	10

Table 3
Comparison training and testing accuracy (error) of ELM and EELM.

Data sets	ELM		EELM	
	Training	Testing	Training	Testing
Diabetes	0.7703	0.7772	0.6767	0.6881
Statlog	0.8159	0.8125	0.4711	0.4652
Glass ID	0.9483	0.4219	0.9109	0.4489
Housing	7.4639	1.1198×10^{10} 326.5156 (best)	22.7959	15.3878
Slump	7.9927	3.2439×10^{5} 8.8406 (best)	17.8965	19.1654

Table 4
Comparison of training and testing RMSE of ELM and EELM.

Data sets	ELM		EELM	
	Training	Testing	Training	Testing
Diabetes	0.0104	0.0288	0.0252	0.0374
Statlog	0.0114	0.0104	0.0508	0.0525
Glass ID	0.0028	0.0286	0.0339	0.0258
Housing	0.2381	3.6669×10^{10}	0.5161	0.7257
Slump	0.1394	2.1991×10^{6}	0.5151	0.6818

In the regression cases, EELM has a better robustness property than ELM. Figs. 3–5 show that EELM is more steady than ELM for regression cases.

Table 5
Comparison of average training and testing time of ELM and EELM.

Data sets	ELM (s)		EELM (s)	
	Training	Testing	Training	Testing
Diabetes	0.0034	0.0012	0.0094	0.0012
Statlog	0.0208	0.0208	0.0271	0.0083
Glass ID	0.0012	0.0006	0.0246	0.0012
Housing	0.0508	0.0117	0.0258	0.0023
Slump	6.2500×10^{-4}	0.0022	0.0075	0.0028

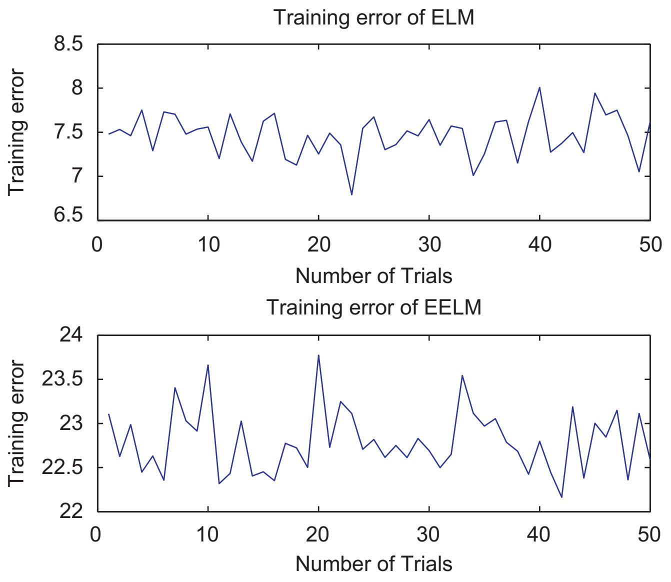

Fig. 3. Training accuracy of two algorithms for Housing.

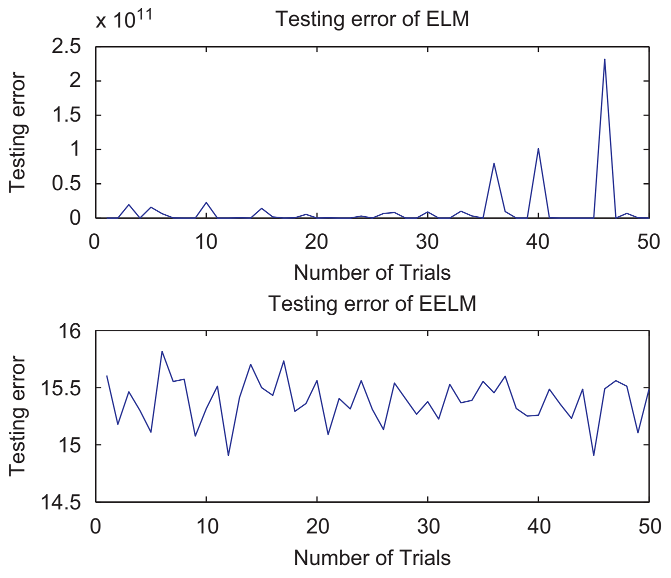

Fig. 4. Testing accuracy of two algorithms for Housing.

In the Diabetes case, the performance of both ELM and EELM including training accuracy, testing accuracy and learning time of two algorithms for 25 different SFLNs with 20–500 nodes were measured and the results are reported in Figs. 6–8, which show that in the simulation of the mid size classification application, ELM

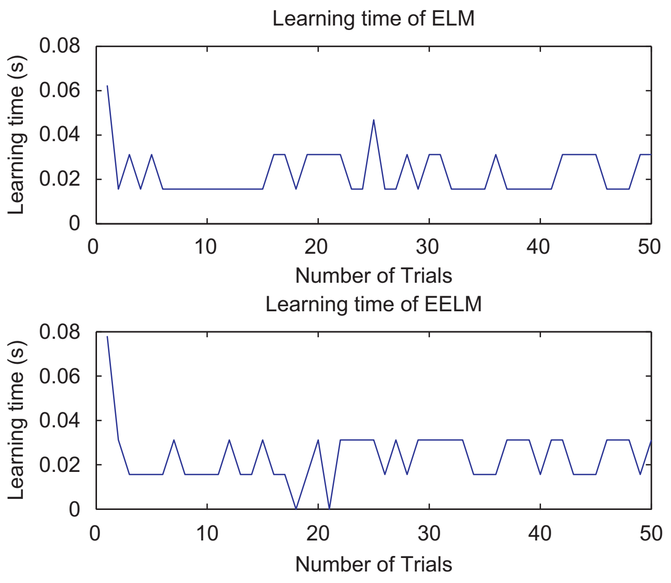

Fig. 5. Learning time of two algorithms for Housing.

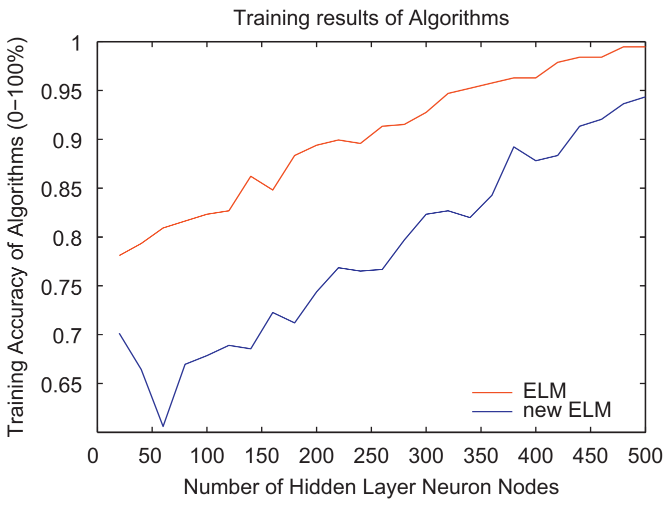

Fig. 6. Training accuracy of two algorithms for Diabetes.

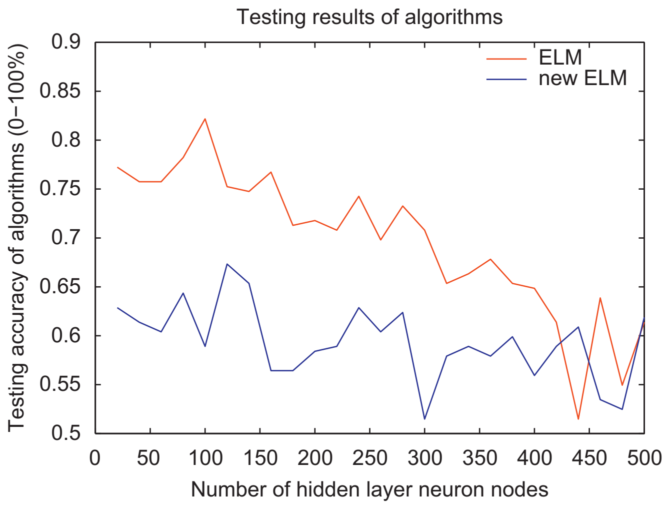

Fig. 7. Testing accuracy of two algorithms for Diabetes.

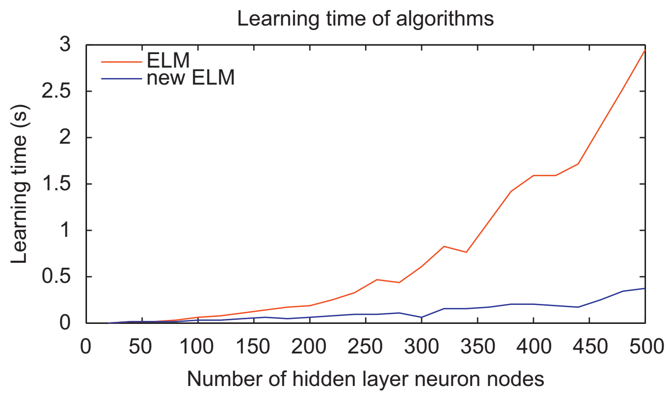

Fig. 8. Learning time of two algorithms for Diabetes.

can reach a higher testing rate than EELM with same number of nodes. Whereas, the time spent by ELM increases much faster than that spent by EELM with the increasing of the number of nodes.

5. Discussions and conclusions

This paper proposed a simple and effective algorithm for single-hidden layer feedforward neural networks (SLFNs) called effective extreme learning machine (EELM) in an attempt to overcome the shortcomings of the extreme learning machine (ELM). There are several interesting features of the proposed EELM algorithm in comparison with the ELM algorithm:

(1) The learning speed of EELM is generally faster than ELM. The main difference between EELM and ELM algorithms lie in the selection of input weights and biases. The ELM algorithm chooses them randomly which consumes little time. Our EELM algorithm selects the input weights and biases properly, which also consumes short time compared with the training time of output weights.
(2) The proposed EELM by making proper selection of input weights and biases of the neural networks avoids the risk of yielding singular or not full column rank hidden layer output matrix **H**. This allows for the use of a faster method which can calculate the Moore–Penrose generalized inverse of **H** much more rapidly.
(3) Another impressive feature the EELM possesses is that it has a longer prediction term with acceptable accuracy than the ELM algorithm. Also, EELM has better robustness property than ELM. In particular, in the regression, the performance of ELM is sometimes poor. But EELM remains steady and has a good performance.

It is worthwhile pointing that in our algorithm in order to sort the samples by affine transformation $X \mapsto W \cdot X + b$, we adopt the method of decimal numeral system. However, when there is high-dimensional data and the range of the deviation between samples $|x_{i+1,j} - x_{ij}|$ (the symbols here have the same meanings as in Section 3) is very big, the weights $w_j^2 = w_j^1 10^{\sum_{p=1}^{j} n_p}$ become so large that the computer will treat it as infinity. To resolve the problem, one can use algorithms of large number operation. Whether there exist better methods to sort high-dimensional data effectively and simply by an affine transformation keeps open.

Finally, the proposed EELM algorithm is effective when the activation functions satisfy the assumptions in Theorem 2.3. Gaussian radial basis function belongs to this kind of functions. Nonetheless, the sigmoidal function is not included. This poses a new problem of designing algorithms using other kinds of activation functions, which are as effective and fast as EELM.

References

[1] P.L. Bartlett, The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network, IEEE Transactions on Information Theory 44 (2) (1998) 525–536.
[2] C. Blake, C. Merz, UCI repository of machine learning databases, Department of Information and Computer Sciences, University of California, Irvine, USA, 1998, ⟨http://www.ics.uci.edu/~ mlearn/MLRepository.html⟩.
[3] F.L. Cao, T.F. Xie, Z.B. Xu, The estimate for approximation error of neural networks: a constructive approach, Neurocomputing 71 (2008) 626–630.
[4] T.P. Chen, Approximation problems in system identification with neural networks, Science in China (series A) 37 (4) (1994) 414–421.
[5] T.P. Chen, H. Chen, Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks, IEEE Transactions on Neural Networks 6 (1995) 904–910.
[6] T.P. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Transactions on Neural Networks 6 (1995) 911–917.
[7] G. Cybenko, Approximation by superpositions of sigmoidal function, Mathematics of Control, Signals, and Systems 2 (1989) 303–314.
[8] G. Feng, G.B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks 20 (8) (2009) 1352–1357.
[9] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: International Conference on Machine Learning1996, pp. 148–156.
[10] K.I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (1989) 183–192.
[11] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks 4 (1991) 251–257.
[12] G.B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Transactions on Neural Networks 14 (2) (2003) 274–281.
[13] G.B. Huang, H.A. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions, IEEE Transactions on Neural Networks 9 (1) (1998) 224–229.
[14] G.B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.
[15] G.B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (2008) 3460–3468.
[16] G.B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.
[17] G.B. Huang, X.J. Ding, H.M. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74 (1–3) (2010) 155–163.
[18] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[19] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks 6 (1993) 861–867.
[20] M.B. Li, G.B. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing 68 (2005) 306–314.
[21] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Applications, Wiley, New York, 1972.
[22] G. Rätsch, T. Onoda, K.R. Müller, An improvement of AdaBoost to avoid overfitting, in: Proceedings of the Fifth International Conference on Neural Information Processing (ICONIP' 1998)1998.
[23] E. Romero, R. Alquézar, A new incremental method for function approximation using feed-forward neural networks, in: Proceedings of INNS-IEEE International Joint Conference on Neural Networks (IJCNN2002)2002, pp. 1968–1973.
[24] H.J. Rong, G.B. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 39 (4) (2009) 1067–1072.
[25] D. Serre, Matrices: Theory and Applications, Springer, New York, 2010.
[26] D.R. Wilson, T.R. Martinez, Heterogeneous radial basis function networks, in: Proceedings of the International Conference on Neural Networks (ICNN 96) 1996, pp. 1263–1267.
[27] R. Zhang, G.B. Huang, N. Sundararajan, P. Saratchandran, Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis, IEEE/ACM Transactions on Computational Biology and Bioinformatics 4 (3) (2007) 485–495.
[28] Q.Y. Zhu, A.K. Qin, P.N. Suganthan, G.B. Huang, Evolutionary extreme learning machine, Pattern Recognition 38 (10) (2005) 1759–1763.

Yuguang Wang received the B.Sc. degree in Mathematics and its Application from China Jiliang University, China, in 2008. He is currently working towards the M.Sc. degree from China Jiliang University, China. His research interests include artificial neural networks and harmonic analysis and approximation on the unit sphere.

Feilong Cao was born in Zhejiang Province, China, on August 1965. He received the B.S. degree in Mathematics and the M.S. degree in Applied Mathematics from Ningxia University, China, in 1987 and 1998, respectively. In 2003, he received the Ph.D. degree in Institute for Information and System Science, Xi'an Jiaotong University, China. From 1987 to 1992, he was an assistant professor. He is now a professor in China Jiliang University. His current research interests include neural networks and approximation theory. He is the author or coauthor of more than 100 scientific papers.

Yubo Yuan received the B.Sc. and M.Sc. degrees in Information and Computational Science from Lanzhou University, PR China, in 1997 and 2000, and Ph.D. degree Information and Computational Science from Xi'an Jiaotong University, PR China, in 2003. From November 2003 to June 2006, he worked as lecture and associate professor in University of Electronic Science and Technology of China. From July 2006 to August 2008, he worked as research assistant professor in mathematics department of Virginia Tech, USA. From February 2009, he has been working as director in the Institute of Metrology and Computational Science, China Jiliang University, PR China. His current research includes optimization, data mining, machine learning, computational intelligence and extreme learning machine.