Neural Comput & Applic (2013) 22:541–550
DOI 10.1007/s00521-012-0860-2

A modified extreme learning machine with sigmoidal activation functions

Zhixiang X. Chen · Houying Y. Zhu · Yuguang G. Wang

Received: 20 September 2011 / Accepted: 21 January 2012 / Published online: 15 February 2012

Abstract This paper proposes a modified ELM algorithm that properly selects the input weights and biases before training the output weights of single-hidden layer feedforward neural networks with sigmoidal activation function and proves mathematically the hidden layer output matrix maintains full column rank. The modified ELM avoids the randomness compared with the ELM. The experimental results of both regression and classification problems show good performance of the modified ELM algorithm.

Keywords Feedforward neural networks · Extreme learning machine · Moore–Penrose generalized inverse

1 Introduction

Feedforward neural networks have been deeply and systematically studied because of their universal approximation capabilities on compact input sets and approximation in a finite set. Cybenko [1] and Funahashi [2] proved that any continuous function could be approximated on a compact set with uniform topology by a single-hidden layer feedforward neural network (SLFN) with any continuous sigmoidal activation function. Hornik in [3] showed that any measurable function could be approached with such networks. A variety of results on SLFN approximations to multivariate functions were later established by many authors: [4–9] by Cao, Xu et al. [10], [11] by Chen and Chen, [12] by Hahm and Hong, [13] by Leshno et al., etc. SLFNs have been extensively used in many fields due to their abilities: (1) to approximate complex nonlinear mappings directly from the input samples; and (2) to provide models for a large class of natural and artificial phenomena that are difficult to handle with using classical parametric methods.

It is known that the problems of density and complexity in neural network are theoretical bases for algorithms. In practical applications, it is paid close attention to the faster learning algorithms of neural networks in a finite training set.

Huang and Babri [14] showed that a single-hidden layer feedforward neural network with at most N hidden nodes and with almost any type of nonlinear activation function could exactly learn N distinct samples. Although conventional gradient-based learning algorithms, such as backpropagation (BP) and its variant Levenberg–Marquardt (LM) method, have been extensively used in training multilayer feedforward neural networks, these learning algorithms are still relatively slow and may also easily get stuck in a local minimum. Support vector machines (SVMs) have been extensively used for complex mappings and famous for its good generalization ability. However, to tune SVM kernel parameters finely is a time-intensive business.

This paper is organized as follows. Section 2 gives theoretical deductions of modified ELM algorithm with sigmoidal activation function. Section 3 proposes a modified ELM algorithm. Section 4 presents performance evaluation. Section 5 consists of discussions and conclusions.

Z. X. Chen
Department of Mathematics, Shaoxing University, Shaoxing 312000, Zhejiang Province, People's Republic of China

H. Y. Zhu · Y. G. Wang (✉)
Department of Information and Mathematics Sciences, China Jiliang University, Hangzhou 310018, Zhejiang Province, People's Republic of China
e-mail: wangyg85@gmail.com

2 Theoretical deductions of modified ELM algorithm with sigmoidal activation function

Recently, a new learning algorithm for SLFN named extreme learning machine (ELM) has been proposed by Huang et al. in [15, 16]. Unlike traditional approaches (such as BP algorithms, SVMs), ELM algorithm has a concise architecture, no need to tune input weights and biases. Particularly, the learning speed of ELM can be thousands of times faster than traditional feedforward network learning algorithms like BP algorithm. Compared with traditional learning algorithms, ELM not only tends to reach the smallest training error but also to obtain the smallest norm of weights. According to Bartlett in [17], the smaller training error the neural network reaches and the smaller the norm of weights is, the better generalization performance the networks tend to have. Therefore, the ELM can have good generalization performance. So far much work has been conducted on ELM and its related problems, and some relevant results have been obtained in [18–22].

The ELM algorithm of SLFNs can be summarized as the following three steps:

Algorithm of 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.

Step 1: Randomly assign input weight W_i and bias b_i $(i = 1, 2, \ldots, N)$

Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.

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

Several methods, such as orthogonal projection method, iterative method and singular value decomposition (SVD), can be used to compose the Moore–Penrose generalized inverse of $\mathbf{H}$. The orthogonal projection, orthogonalization method and iterative method have their limitations as searching and iteration may consume extra training time. The orthogonal project method can be used when the hidden layer output matrix is a full column rank matrix. SVD can be generally used to calculate the Moore–Penrose generalized inverse of hidden layer output matrix in all cases, but it costs plenty of time. This paper designs a learning machine that first properly trains input weights and biases such that the hidden layer output matrix is full column rank, and then orthogonal project method will be used to calculate the Moore–Penrose generalized inverse of hidden layer output matrix.

For n arbitrary distinct sample (X_i, t_i), 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 hidden nodes and activation function $g(x)$ are mathematically modeled as

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

where $\beta_i = (\beta_{i1}, \beta_{i2}, \ldots, \beta_{iN})^T$ is the output weight vector connecting the ith nodes and the output nodes, $W_i = (w_{i1}, w_{i2}, \ldots, w_{id})^T$ is the input weight vector connecting the ith hidden nodes and the input nodes, and b_i is the threshold of the ith hidden node. The SLFNs $G_N(x)$ can approximate these n samples with zero error means

$$G_N(x_j) = t_j, \quad j = 1, 2, \ldots, N. \tag{2}$$

The above n equations can be written as

$$H\beta = T, \tag{3}$$

where

$$H = H(w_1, w_2, \ldots, w_N; b_1, b_2, \ldots, b_N; X_1, X_2, \ldots, X_n) = \begin{pmatrix} g(W_1 \cdot X_1 + b_1) & g(W_2 \cdot X_1 + b_2) & \cdots & g(W_N \cdot X_1 + b_N) \\ g(W_1 \cdot X_2 + b_1) & g(W_2 \cdot X_2 + b_2) & \cdots & g(W_N \cdot X_2 + b_N) \\ \vdots & \vdots & \cdots & \vdots \\ g(W_1 \cdot X_n + b_1) & g(W_2 \cdot X_n + b_2) & \cdots & g(W_N \cdot X_n + b_N) \end{pmatrix}_{n \times N}, \tag{4}$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_N^T \end{pmatrix}_{N \times m} \quad \text{and} \quad T = \begin{pmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_n^T \end{pmatrix}_{n \times m}. \tag{5}$$

As named in Huang et al. [14–23], H is called the hidden layer output matrix of the neural network.

In most cases, the number of hidden nodes is much less than the number of training samples, $N \ll n$. Then, there may not exist $w_i, b_i, \beta_i (i = 1, 2, \ldots, N)$ such that $H\beta = T$. So, the least-square method is used to find the least-square solutions of $H\beta = T$. Furthermore, the special solution $\hat{\beta}$ can be computed,

$$\|H\hat{\beta} - T\| = \min_{\beta} \|H\beta - T\|, \tag{6}$$

this $\hat{\beta}$ is the minimum norm least-square solution. From [24, 27],

$$\hat{\beta} = H^{\dagger}T, \tag{7}$$

where $H^{\dagger}$ is called the Moore–Penrose generalized inverse of matrix of H. Thus, the output weight of ELM algorithm is

$$\beta = H^{\dagger}T. \tag{8}$$

Suppose $X_1, X_2 \in \mathbb{R}^d$, $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$, $i = 1, 2$. We say $X_1 \prec X_2$ if and only if there exists $j_0 \in$

$\{1,2,\ldots,d\}$ such that $x_{1j_0} < x_{2j_0}$, $x_{1j} = x_{2j}$, $j = j_0+1, j_0+2,\ldots,d$.

Lemma 2.1 *For n distinct vectors $X_1 \prec X_2 \prec \cdots \prec X_n \in \mathbb{R}^d (d \geq 2)$, there exists a vector $W \in \mathbb{R}^d$ such that*

$$W \cdot X_1 < W \cdot X_2 < \cdots < W \cdot X_n. \tag{9}$$

Proof For $X_i = (x_{i1}, x_{i2}, \ldots, x_{id}), i = 1,2,\ldots,n$, we set

$$w_j^1 := \frac{1}{1+\max_i\{|x_{ij}|\}}, \quad j = 1,2,\ldots,d, \tag{10}$$

then

$$x_{ij}^1 = w_j^1 x_{ij} \in [-1,1], \quad i = 1,2,\ldots,n, \quad j = 1,2,\ldots,d. \tag{11}$$

Let

$$y_{ij}^1 = |x_{i+1,j}^1 - x_{ij}^1|, \quad i = 1,2,\ldots,n-1, \quad j = 1,2,\ldots,d. \tag{12}$$

For given $j (1 \leq j \leq d)$, if $y_{ij}^1 = 0$ for all $i = 1,2,\ldots,n-1$, let $n_j = 2d$; otherwise,

$$n_j = \frac{4d}{\min_{y_{ij}^1 \neq 0}\{y_{ij}^1\}}. \tag{13}$$

Obviously,

$$n_j \geq 2d, \quad j = 1,2,\ldots,d. \tag{14}$$

Define

$$w_j^2 := w_j^1 \Pi_{i=1}^j n_i, \quad j = 1,2,\ldots,d, \tag{15}$$

and

$$W = (w_1^2, w_2^2, \ldots, w_d^2). \tag{16}$$

For fixed $i (1 \leq i \leq n-1)$, owing to $X_i \prec X_{i+1}$, there exists $k_0 \in \mathbb{N}$ such that

$$x_{ik_0} < x_{i+1,k_0}; \quad x_{ij} = x_{i+1,j}, \quad j = k_0+1,\ldots,d. \tag{17}$$

Write

$$\begin{aligned} W \cdot X_{i+1} - W \cdot X_i &= \sum_{k=1}^{k_0-1}(x_{i+1,k}^1 - x_{i,k}^1)\Pi_{s=1}^k n_s \\ &\quad + (x_{i+1,k_0}^1 - x_{ik_0}^1)\Pi_{s=1}^{k_0} n_s \\ &=: I_1 + I_2. \end{aligned} \tag{18}$$

Since

$$\begin{aligned} |I_1| &\leq 2(n_1 + n_1 n_2 + \cdots + n_1 n_2 \cdots n_{k_0-1}) \\ &= 2\Pi_{s=1}^{k_0-1} n_s \left(1 + \frac{1}{n_{k_0-2}} + \cdots + \frac{1}{n_2 \cdots n_{k_0-1}}\right) \\ &\leq 2\Pi_{s=1}^{k_0-1} n_s \cdot \frac{1}{1-\frac{1}{2d}} < 2d\Pi_{s=1}^{k_0-1} n_s. \end{aligned} \tag{19}$$

And

$$\begin{aligned} I_2 &= \Pi_{s=1}^{k_0-1} n_s \cdot n_{k_0}(x_{i+1,k_0}^1 - x_{ik_0}^1) \\ &= \Pi_{s=1}^{k_0-1} n_s \cdot n_{k_0} \cdot y_{i,k_0}^1 \\ &\geq \Pi_{s=1}^{k_0-1} n_s \cdot y_{i,k_0}^1 \cdot \frac{4d}{y_{ik_0}^1} \\ &= 4d\Pi_{s=1}^{k_0-1} n_s. \end{aligned} \tag{20}$$

This gives $W \cdot X_{i+1} - W \cdot X_i > 0$, that is, $W \cdot X_i < W \cdot X_{i+1}$, which completes the proof of Lemma 2.1. □

Remark 1 From Lemma 2.1, it is deduced that the sequence of the samples is unchanged under the affine transformation $X_i \mapsto W \cdot X_i$. It is noteworthy that Lemma 2.1 also provides the method (10)–(16) to compute the weight W of the transformation, and the method here is better than that given in [22] as the weights in the Lemma 2.1 are much smaller than those in [22]. With this property, it is possible to make the hidden layer output matrix full column rank.

Suppose that $\sigma(x)$ is a bounded sigmoidal function, then

$$\lim_{x\to+\infty} \sigma(x) = 1, \quad \lim_{x\to-\infty} \sigma(x) = 0. \tag{21}$$

Set

$$\delta_\sigma(A) := \sup_{x \geq A} \max\{|1-\sigma(x)|, |\sigma(-x)|\}, \tag{22}$$

then

$$\lim_{A\to+\infty} \delta_\sigma(A) = 0. \tag{23}$$

Let $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$ be a set of samples, and $x_1 < x_2 < \cdots < x_n$. We have the following lemma.

Lemma 2.2 *There exist numbers $w_1, w_2, \ldots, w_n$ and $\alpha_1, \alpha_2, \ldots, \alpha_n$ such that the matrix*

$$G_\sigma = \begin{pmatrix} \sigma(w_1x_1+\alpha_1) & \sigma(w_2x_1+\alpha_2) & \cdots & \sigma(w_nx_1+\alpha_n) \\ \sigma(w_1x_2+\alpha_1) & \sigma(w_2x_2+\alpha_2) & \cdots & \sigma(w_nx_2+\alpha_n) \\ \vdots & \vdots & \cdots & \vdots \\ \sigma(w_1x_n+\alpha_1) & \sigma(w_2x_n+\alpha_2) & \cdots & \sigma(w_nx_n+\alpha_n) \end{pmatrix} \tag{24}$$

is nonsingular.

Proof Since

$$\lim_{A\to+\infty} \delta_\sigma(A) = 0, \tag{25}$$

there exists $A > 0$ such that $\delta_\sigma(A) < 1/(4n)$. Hence,

$$|1-\sigma(A)| < \frac{1}{4n}, \quad |\sigma(-A)| < \frac{1}{4n}, \tag{26}$$

and

$$|\sigma(c)| < \frac{1}{4n}, \quad |1-\sigma(-c)| < \frac{1}{4n}, \quad c < -A. \tag{27}$$

Input weights and biases are chosen by the following criteria.

$$w_1 = -\frac{2A}{x_2 - x_1}, \quad w_2 = -\frac{2A}{x_3 - x_2}, \quad \ldots,$$
$$w_{n-1} = -\frac{2A}{x_n - x_{n-1}}, \quad w_n = -\frac{2A}{x_n - x_{n-1}},$$
$$\alpha_1 = A + \frac{2Ax_1}{x_2 - x_1}, \quad \alpha_2 = A + \frac{2Ax_2}{x_3 - x_2}, \quad \ldots,$$
$$\alpha_{n-1} = A + \frac{2Ax_{n-1}}{x_n - x_{n-1}}, \quad \alpha_n = A + \frac{2Ax_n}{x_n - x_{n-1}},$$

it follows that

$$G_\sigma = \begin{pmatrix} \sigma(A) & \sigma\left(-\frac{2A(x_1-x_2)}{x_3-x_2}+A\right) & \cdots & \sigma\left(-\frac{2A(x_1-x_{n-1})}{x_n-x_{n-1}}+A\right) & \sigma\left(-\frac{2A(x_1-x_n)}{x_n-x_{n-1}}+A\right) \\ \sigma(-A) & \sigma(A) & \cdots & \sigma\left(-\frac{2A(x_2-x_{n-1})}{x_n-x_{n-1}}+A\right) & \sigma\left(-\frac{2A(x_2-x_n)}{x_n-x_{n-1}}+A\right) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \sigma\left(-\frac{2A(x_{n-1}-x_1)}{x_2-x_1}+A\right) & \sigma\left(-\frac{2A(x_{n-1}-x_2)}{x_3-x_2}+A\right) & \cdots & \sigma(A) & \sigma\left(-\frac{2A(x_{n-1}-x_n)}{x_n-x_{n-1}}+A\right) \\ \sigma\left(-\frac{2A(x_n-x_1)}{x_2-x_1}+A\right) & \sigma\left(-\frac{2A(x_n-x_2)}{x_3-x_2}+A\right) & \cdots & \sigma(-A) & \sigma(A) \end{pmatrix}. \tag{28}$$

In turn, subtract the second row from the first row, the third row from the third row, ……, and keep the last row unchanged. Denote the matrix that has performed the above row operations by $\widetilde{G}_\sigma = (\widetilde{g}_{ij})_n$.

Now for $i = 1, 2, \ldots, n-1$, the elements of the ith row can be estimated as follows. For $j = 1, \ldots, i-1$, since

$$-\frac{2A(x_i - x_j)}{x_{j+1} - x_j} + A < -A, \tag{29}$$

$$|\widetilde{g}_{ij}| = \left|\sigma\left(-\frac{2A(x_i - x_j)}{x_{j+1} - x_j} + A\right) - \sigma\left(-\frac{2A(x_{i+1} - x_j)}{x_{j+1} - x_j} + A\right)\right| < \frac{1}{2n}. \tag{30}$$

For $j = i$,

$$|\widetilde{g}_{ii}| = |\sigma(A) - \sigma(-A)| \geq 1 - (|1 - \sigma(A)| + |\sigma(-A)|) > 1 - \frac{1}{2n}. \tag{31}$$

For $j = i+1, \ldots, n-1$,

$$-\frac{2A(x_i - x_j)}{x_{j+1} - x_j} + A > A, \tag{32}$$

implies

$$|\widetilde{g}_{ij}| = \left|\sigma\left(-\frac{2A(x_i - x_j)}{x_{j+1} - x_j} + A\right) - \sigma\left(-\frac{2A(x_{i+1} - x_j)}{x_{j+1} - x_j} + A\right)\right| \leq \left|1 - \sigma\left(-\frac{2A(x_i - x_j)}{x_{j+1} - x_j} + A\right)\right| + \left|1 - \sigma\left(-\frac{2A(x_{i+1} - x_j)}{x_{j+1} - x_j} + A\right)\right| < \frac{1}{2n}. \tag{34}$$

For $j = n$, it can similarly be obtained that

$$|\widetilde{g}_{in}| = \left|\sigma\left(-\frac{2A(x_i - x_n)}{x_n - x_{n-1}} + A\right) - \sigma\left(-\frac{2A(x_{i+1} - x_n)}{x_n - x_{n-1}} + A\right)\right| < \frac{1}{2n}, \tag{35}$$

In the nth row of $\widetilde{G}_\sigma$, for $j = 1, 2, \ldots, n-1$,

$$-\frac{2A(x_n - x_j)}{x_{j+1} - x_j} + A < -A \tag{36}$$

which implies

$$|\widetilde{g}_{nj}| < \frac{1}{4n} \tag{37}$$

and

$$\widetilde{g}_{nn} = \sigma(A) > 1 - \frac{1}{4n}. \tag{38}$$

Therefore,

$$|\widetilde{g}_{ii}| > \sum_{1 \leq i \neq j \leq n} |\widetilde{g}_{ij}|, \quad i = 1, 2, \ldots, n, \tag{39}$$

that is, $\widetilde{G}_\sigma$ is strictly diagonally dominant which guarantees that $\widetilde{G}_\sigma$ is nonsingular. This indicates the nonsingularity of G_σ. □

Remark 2 Lemma 2.2 gives the method to construct the weights and biases of the single-hidden layer neural network with the sigmoidal activation function such that the hidden layer output matrix is a full column rank for any given data. With Lemma 2.2, it is possible to construct the algorithm to train the neural network.

For $X_i \in \mathbb{R}^d$, $i = 1, 2, \ldots, N$, and $X_1 \prec X_2 \prec \cdots \prec X_N$. Lemma 1 illustrates that there exists $W \in \mathbb{R}^d$, such that

$$W \cdot X_1 < W \cdot X_2 < \cdots < W \cdot X_N. \tag{40}$$

Let $x_i = W \cdot X_i,\ i = 1, 2, \ldots, N$, and set

$$W_1 = -\frac{2A}{x_2 - x_1} W, \quad W_2 = -\frac{2A}{x_3 - x_2} W, \quad \ldots,$$
$$W_{N-1} = -\frac{2A}{x_N - x_{N-1}} W, \quad W_N = -\frac{2A}{x_N - x_{N-1}} W.$$
$$b_1 = A + \frac{2Ax_1}{x_2 - x_1}, \quad b_2 = A + \frac{2Ax_2}{x_3 - x_2}, \quad \ldots,$$
$$b_{N-1} = A + \frac{2Ax_{N-1}}{x_N - x_{N-1}}, \quad b_N = A + \frac{2Ax_N}{x_N - x_{N-1}},$$

where A satisfies $\delta_\sigma(A) < 1/(4N)$.

From Lemma 2.2, the following theorem is easily obtained.

Theorem 2.1 *Suppose $\sigma(x)$ is a bounded sigmoidal function, $X_i \in \mathbb{R}^d,\ i = 1, 2, \ldots, N$, and $X_1 \prec X_2 \prec \cdots \prec X_N$. Then, there exist $W_i \in \mathbb{R}^d,\ b_i \in \mathbb{R},\ i = 1, 2, \ldots, N$, such that the matrix*

$$H = \begin{pmatrix} \sigma(W_1 \cdot X_1 + b_1) & \sigma(W_2 \cdot X_1 + b_2) \cdots \sigma(W_N \cdot X_1 + b_N) \\ \sigma(W_1 \cdot X_2 + b_1) & \sigma(W_2 \cdot X_2 + b_2) \cdots \sigma(W_N \cdot X_2 + b_N) \\ & \vdots \cdots \vdots \\ \sigma(W_1 \cdot X_N + b_1) & \sigma(W_2 \cdot X_N + b_2) \cdots \sigma(W_N \cdot X_N + b_N) \end{pmatrix} \tag{41}$$

is nonsingular.

Based on the knowledge of matrix and Theorem 2.1, we have the following conclusion.

Corollary 2.1 *Suppose $X_i \in \mathbb{R}^d,\ i = 1, 2, \ldots, n$ are n distinct vectors, $N > n$. Then, there exist $W_i \in \mathbb{R}^d, b_i \in \mathbb{R}$, $i = 1, 2, \ldots, N$ such that the matrix*

$$H = \begin{pmatrix} \sigma(W_1 \cdot X_1 + b_1) & \sigma(W_2 \cdot X_1 + b_2) & \cdots & \sigma(W_N \cdot X_1 + b_N) \\ \sigma(W_1 \cdot X_2 + b_1) & \sigma(W_2 \cdot X_2 + b_2) & \cdots & \sigma(W_N \cdot X_2 + b_N) \\ \vdots & \vdots & \cdots & \vdots \\ \sigma(W_1 \cdot X_n + b_1) & \sigma(W_2 \cdot X_n + b_2) & \cdots & \sigma(W_N \cdot X_n + b_N) \end{pmatrix}_{n \times N} \tag{42}$$

is full column rank.

3 A modified ELM using sigmoidal activation function

According to the discussion of Sect. 2 and based on ELM algorithm, a new algorithm is proposed. It uses sigmoidal function as its activation function and can make good use of orthogonal projection method to calculate the output weights. The modified extreme learning machine is stated in the following algorithm.

Algorithm of Modified ELM: 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 sigmoidal function (for instance, $g(x) = 1/(1 + e^{-x})$) and hidden node number N.

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

Sort the former N samples $(X_i^*, t_i^*)\ (i = 1, 2, \ldots, N)$ in terms of $W \cdot X_1^*, W \cdot X_2^*, \ldots, W \cdot X_N^*$ such that $W \cdot X_{i_1}^* < W \cdot X_{i_2}^* < \cdots < W \cdot X_{i_N}^*$ ($i_j \neq i_k$ for $j \neq k, j, k = 1, 2, \ldots, N$ and $i_j = 1, 2, \ldots, N$) and denote the sorted data as $(X_i, t_i)\ (i = 1, 2, \ldots, n)$ and $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})\ (i = 1, 2, \ldots, n)$. For $j = 1, 2, \ldots, d$, make following calculations.

$$w_j^1 = \frac{1}{\max_{i=1,2,\ldots,N}\{|x_{ij}|\}} > 0, \tag{43}$$

$$x_{ij}^1 = w_j^1 x_{ij} \in [-1, 1], \tag{44}$$

$$y_{ij}^1 = \left| x_{i+1,j}^1 - x_{ij}^1 \right| \quad (i = 1, 2, \ldots, N-1), \tag{45}$$

$$n_j = \begin{cases} 2d, & \text{if } y_{ij}^1 = 0 \text{ for all } i = 1, 2, \ldots, N-1, \\ \frac{4d}{\min_{y_{ij}^1 \neq 0}\{y_{ij}^1\}}, & \text{else,} \end{cases} \tag{46}$$

$$w_j^2 = w_j^1 \Pi_{i=1}^j n_i. \tag{47}$$

Set

$$W = \left(w_1^2, w_2^2, \ldots, w_d^2\right). \tag{48}$$

Let $a_i = W \cdot X_i$,

$$W_i = \frac{2A}{a_i - a_i} W, \quad i = 1, 2, \ldots, N-1, \tag{49}$$

$$W_N = W_{N-1}, \tag{50}$$

$$b_i = A + \frac{2Aa_i}{a_{i+1} - a_i}, \quad i = 1, 2, \ldots, N-1, \tag{51}$$

$$b_N = A + \frac{2Aa_N}{a_N - a_{N-1}}. \tag{52}$$

Step 2: Calculate output weights $\beta = (\beta_1, \beta_2, \ldots, \beta_N)$.

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

$$\mathbf{H} = \begin{pmatrix} \sigma(W_1 \cdot X_1 + b_1) & \sigma(W_2 \cdot X_1 + b_2) & \ldots & \sigma(W_N \cdot X_1 + b_N) \\ \vdots & \vdots & \ldots & \vdots \\ \sigma(W_1 \cdot X_N + b_1) & \sigma(W_2 \cdot X_N + b_2) & \ldots & \sigma(W_N \cdot X_N + b_N) \\ \sigma(W_1 \cdot X_{N+1} + b_1) & \sigma(W_2 \cdot X_{N+1} + b_2) & \ldots & \sigma(W_N \cdot X_{N+1} + b_N) \\ \vdots & \vdots & \ldots & \vdots \\ \sigma(W_1 \cdot X_n + b_1) & \sigma(W_2 \cdot X_n + b_2) & \ldots & \sigma(W_N \cdot X_n + b_N) \end{pmatrix}_{n \times N}. \tag{53}$$

Then,

$$\beta = \mathbf{H}^\dagger T = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T T. \tag{54}$$

The ELM proves in practice to be an extremely fast algorithm. This is because it randomly chooses the input weights W_i and biases b_i of the SLFNs instead of carefully selecting. However, this big advantage makes the algorithm less effective sometimes. As discussed in [22], the random selection of input weights and biases is likely to yield an unexpected result that the hidden layer output matrix $\mathbf{H}$ is not full column rank, which might cause two difficulties that cannot be overcome theoretically. First, the SLFNs cannot approximate the training samples with zero error, which relatively lowers the prediction accuracy. Secondly, it is known that the orthogonal projection is typically faster than SVD, and the algorithm proposed here can make good use of the faster method which is unable to be used in ELM due to the requirement of nonsingularity of $\mathbf{H}^T\mathbf{H}$.

4 Simulation results

This section measures the performance of the proposed new ELM learning algorithm. The simulations for ELM and modified ELM algorithms are carried out in the Matlab 7.10 environment running in AMD athlon(tm) II, X3, 425 processor with the speed of 2.71 GHz. The activation function used in algorithm is sigmoid function $g(x) = 1/(1 + e^{-x})$. For classification problems, accuracy rates are used as the measurement of the performance of learning algorithms, that is, the ratio of the number of the testing samples that are correctly classified to the number of all samples. For regression problems, root-mean-square deviation (RMSD) of the difference of target vectors and predicted target vectors by the SFLNs is used. We call this RMSD the error of the algorithm for the testing data. The RMSDs of the accuracy and the error of the algorithms are used as the measurement of fluctuations in algorithms.

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

First of all, the 'SinC' function is used to measure the performance of ELM and the modified ELM algorithms. The target function is as follows.

$$y = f(x) = \begin{cases} \sin(x)/x, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{55}$$

The training set (x_i, y_i) and testing set (u_i, t_i) with 5,000 samples are, respectively, created, where x_i, u_i in training and testing data are distributed in $[-10,10]$, respectively, with uniform step length. Large uniform noise distributed in [0.2, 0.2] has been used in all training data to obtain a real regression problem. The experiment is carried out on these data as follows. There are 20 hidden nodes assigned for both ELM and modified ELM 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. Plus, standard deviation of time and accuracy in the process of training and testing are recorded in Table 2.

It can be seen from Table 1 that the modified ELM algorithm spends 0.0675, 0.0169 s CPU time training and testing, respectively, whereas it takes 0.0541 and 0.0213 s for the ELM algorithm to complete the same process. Consider the accuracy of training and testing of both algorithms. The original ELM performs better than the modified ELM. However, the standard deviations of training and testing time and the accuracy are all smaller than those of ELM as shown in Table 2, which means the modified algorithm is more stable than ELM.

Figure 1 shows the raw data and trained results of ELM algorithm, and Fig. 2 shows the raw data and the approximated results based on the modified ELM algorithm, which also demonstrates that the modified ELM algorithm has an acceptable performance.

4.2 Benchmarking with practical problems applications

Performance comparison of the proposed modified ELM and the ELM algorithms for four real problems is carried out in this section. Classification and regression tasks are included in four real problems that are shown in Table 3. Two of them are classification tasks including Diabetes, Glass Identification (Glass ID), and the other two are regression tasks including Housing and Slump (Concrete Slump). All the data sets are from UCI repository of machine learning databases [28]. The speculation of each database is shown in the Table 3. For the databases that

Table 1 Performance comparison for learning function: SinC

Algorithms	Time		Accuracy		# Nodes
	Training	Testing	Training	Testing	
ELM	0.0675	0.0169	0.1150	0.0090	20
Modified ELM	0.0541	0.0213	0.1393	0.0792	20

Table 2 Standard deviation comparison for learning function: SinC

Algorithms	Time		Accuracy	
	Training	Testing	Training	Testing
ELM	0.0251	0.0202	8.5681×10^{-6}	1.0723×10^{-4}
Modified ELM	0.0431	0.0211	5.6075×10^{-17}	4.2056×10^{-17}

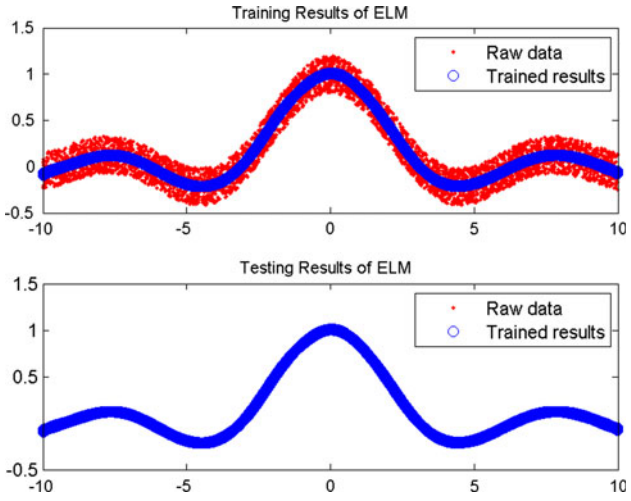

Fig. 1 Outputs of ELM learning algorithm

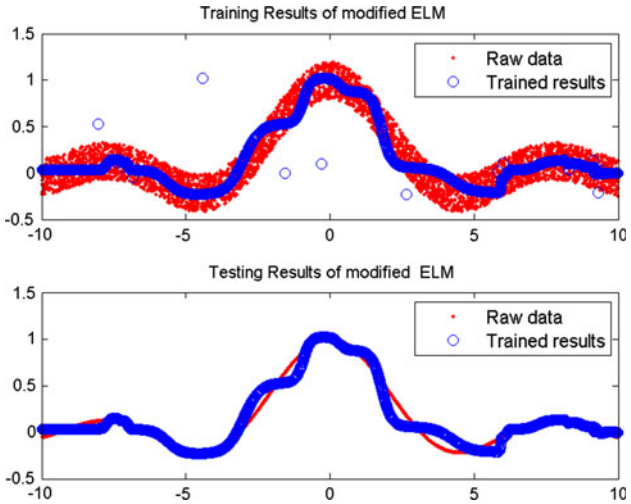

Fig. 2 Outputs of modified ELM learning algorithm

Table 3 Speculations of real-world applications and the number of nodes for each

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

have only one data table, as conducted in [25, 26, 29, 30], 75 and 25% of samples in the problem are randomly chosen for training and testing, respectively, at each trial.

In order to reduce the random error, for every database, fifty trials for both two algorithms are conducted, then taking the average of fifty times as final results. The results are reported in Tables 4, 5 and 6, which show that in our simulation, on the average, both of ELM and the modified ELM algorithms have similar learning time which is reported in Table 6 as well as fast learning rate.

However, the modified ELM has more stable accuracies of training and testing, which can be seen from Table 5, especially in cases of regression problems. From Table 4, it can be observed that the modified ELM algorithm is better than ELM in the accuracy of learning for the regression cases. And detailed information is shown in Figs. 3, 4, 5, 6, 7, 8, 9 and 10.

In addition, it is worth mentioning that compared with ELM algorithm, the modified ELM selects the input weights and biases, which helps avoid the risk of the random errors as we can see from Table 5 and Figs. 3, 4, 5, 6, 7, 8, 9 and 10.

Table 4 Comparison training and testing accuracy/error of ELM and modified ELM

Data sets	ELM		Modified ELM	
	Training	Testing	Training	Testing
Diabetes	0.7784	0.7892	0.6974	0.7327
Glass ID	0.9485	0.4326	0.9375	0.4630
Housing	0.0691	0.0025	0.1214	0.0157
Slump	7.8446	3.4696×10^{7}	7.1422	12.2175

Table 5 Comparison of training and testing RMSD of accuracy/error of ELM and modified ELM

Data sets	ELM		Modified ELM	
	Training	Testing	Training	Testing
Diabetes	0.0068	0.0166	3.3645×10^{-16}	5.6075×10^{-16}
Glass ID	0.0027	0.0264	0	5.6075×10^{-17}
Housing	0.0110	0.0043	9.3181×10^{-17}	0
Slump	0.1995	2.4271×10^{7}	5.3832×10^{-15}	1.2561×10^{-14}

Table 6 Comparison of average training and testing time of ELM and modified ELM

Data sets	ELM		Modified ELM	
	Training	Testing	Training	Testing
Diabetes	0.0084	0.0013	0.0109	0.0037
Glass ID	0.0031	0.0019	0.0178	0.0037
Housing	0.0081	0.0053	0.0088	0.0028
Slump	0.0091	0.0028	0.0163	0.0019

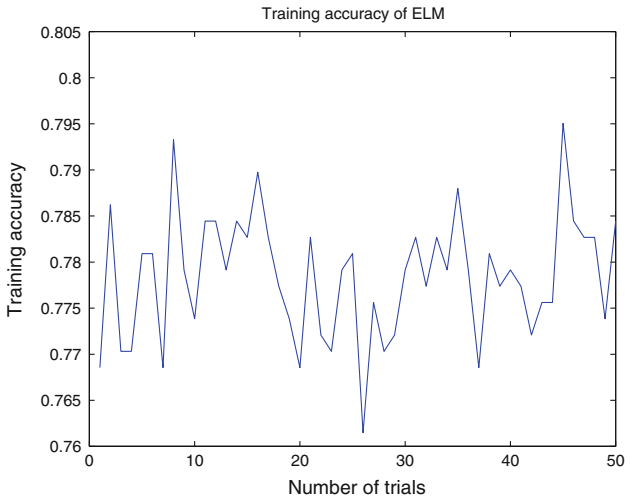

Fig. 3 Training accuracy of ELM for Diabetes

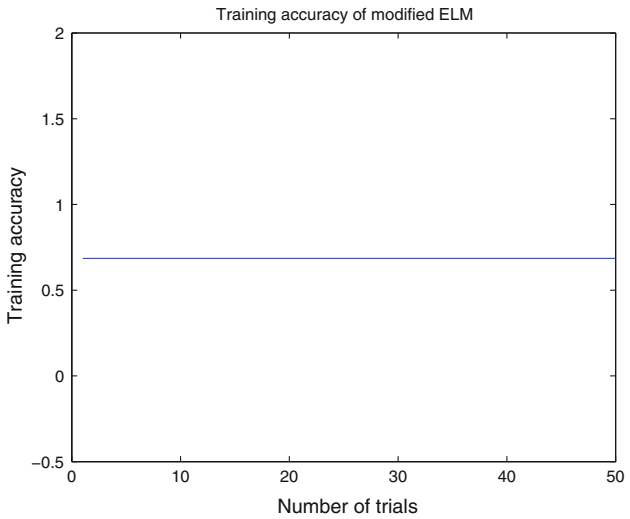

Fig. 4 Training accuracy of modified ELM for Diabetes

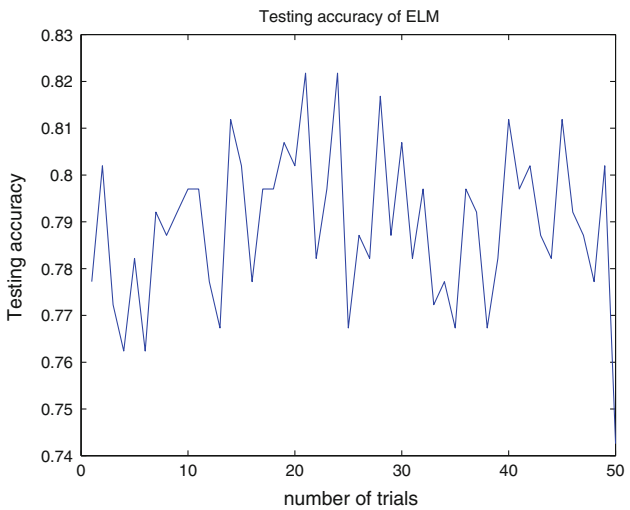

Fig. 5 Testing accuracy of ELM for Diabetes

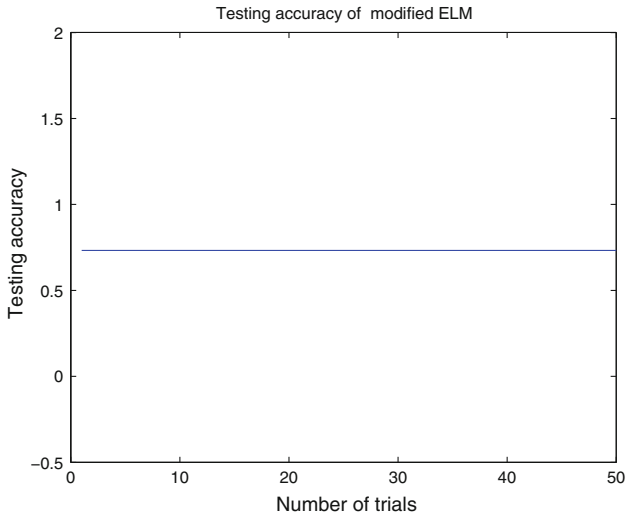

Fig. 6 Testing accuracy of modified ELM for Diabetes

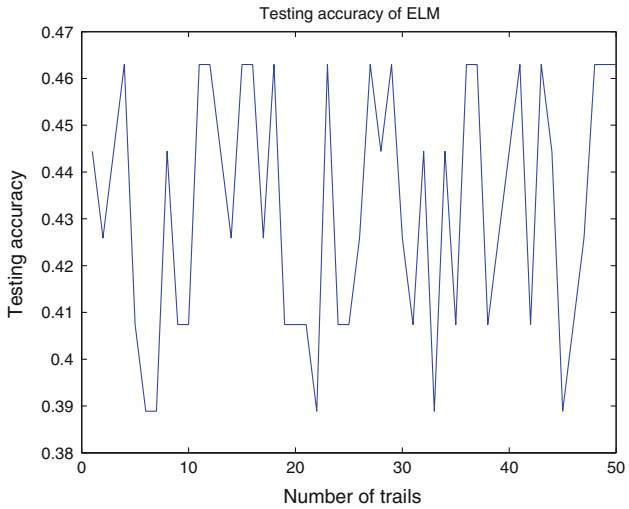

Fig. 7 Testing accuracy of ELM for GlassID

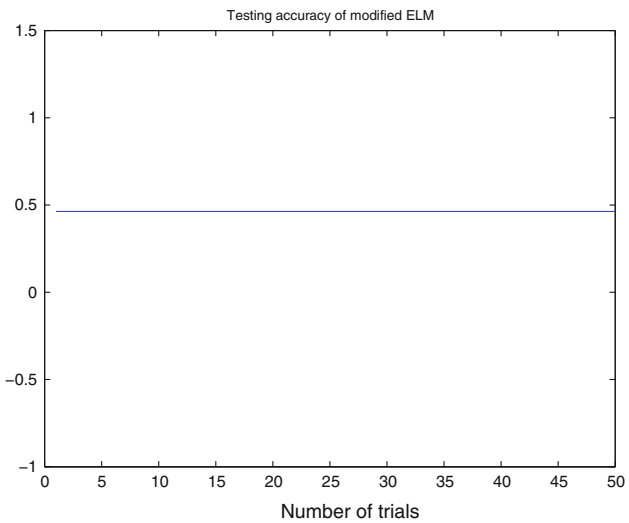

Fig. 8 Testing accuracy of modified ELM for GlassID

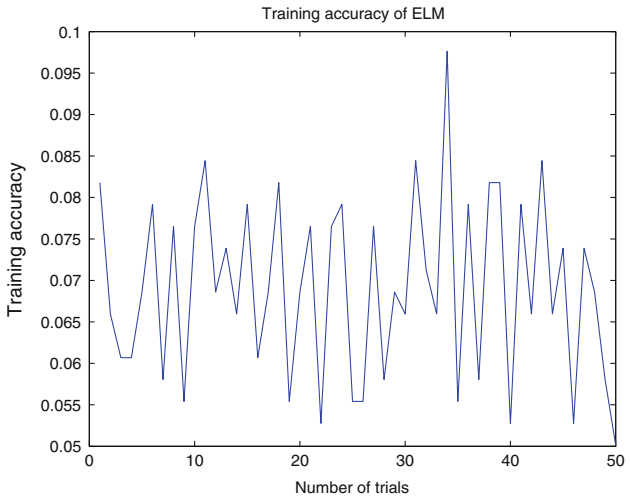

Fig. 9 Training accuracy of ELM for Housing

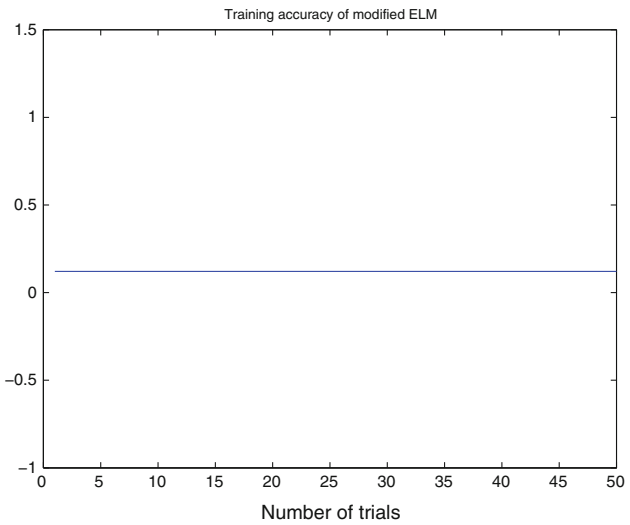

Fig. 10 Training accuracy of modified ELM for Housing

5 Conclusions

This paper proposes a modified ELM algorithm based on the ELM for training single-hidden layer feedforward neural networks (SLFNs) in an attempt to solve the least-square minimization of SLFNs in a more effective way and meanwhile solves the open problem in [22].

The learning speed of modified ELM is as fast as ELM. The main difference between the modified ELM and ELM algorithms lies in the selection of input weights and biases. The modified algorithm selects the input weights and biases properly, which consumes little time compared with the training time of output weights. The modified ELM algorithm overcomes the shortcomings of EELM, the algorithm proposed in [22], that is, it can use sigmoidal function as activation function to train the networks but still keep the qualities of ELM and EELM.

Acknowledgments We would thank Feilong Cao for his suggestions on this paper. The support of the National Natural Science Foundation of China (Nos. 90818020, 10871226, 61179041) is gratefully acknowledged.

References

1. Cybenko G (1989) Approximation by superposition of sigmoidal function. Math Control Signals Syst 2:303–314
2. Funahashi KI (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2:183–192
3. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4:251–257
4. Cao FL, Xie TF, Xu ZB (2008) The estimate for approximation error of neural networks: a constructive approach. Neurocomputing 71:626–630
5. Cao FL, Zhang YQ, He ZR (2009) Interpolation and rate of convergence by a class of neural networks. Appl Math Model 33:1441–1456
6. Cao FL, Zhang R (2009) The errors of approximation for feedforward neural networks in the Lp metric. Math Comput Model 49:1563–1572
7. Cao FL, Lin SB, Xu ZB (2010) Approximation capabilities of interpolation neural networks. Neurocomputing 74:457–460
8. Xu ZB, Cao FL (2004) The essential order of approximation for neural networks. Sci China Ser F Inf Sci 47:97–112
9. Xu ZB, Cao FL (2005) Simultaneous L^p approximation order for neural networks. Neural Netw 18:914–923
10. Chen TP, Chen H (1995) Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. IEEE Trans Neural Netw 6: 904–910
11. Chen TP, Chen H (1995) Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 6:911–917
12. Hahm N, Hong BI (2004) An approximation by neural networks with a fixed weight. Comput Math Appl 47:1897–1903
13. Lan Y, Soh YC, Huang GB (2010, April) Random search enhancement of error minimized extreme learning machine. In: ESANN 2010 proceedings, European symposium on artificial neural networks—computational intelligence and machine learning, pp 327–332
14. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9:224–229
15. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
16. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE international joint conference on neural networks, vol 2, pp 985–990
17. Bartlett PL (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44:525–536
18. Feng G, Huang GB, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20:1352–1357

19. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70:3056–3062
20. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71:3460–3468
21. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17:879–892
22. Wang YG, Cao FL, Yuan YB (2011) A study on effectiveness of extreme learning machine. Neurocomputing 74(16):2483–2490
23. Huang GB (2003) Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw 14:274–281
24. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
25. Rätsch G, Onoda T, Müller KR (1998) An improvement of AdaBoost to avoid overfitting. In: Proceedings of the 5th international conference on neural information processing (ICONIP 1998)
26. Romero E, Alquézar R (2002) A new incremental method for function approximation using feed-forward neural networks. In: Proceedings of the 2002 international joint conference on neural networks (IJCNN'2002), pp 1968–1973
27. Serre D (2000) Matrices: theory and applications. Springer, New York
28. Frank A, Asuncion A (2010) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine. http://archive.ics.uci.edu/ml
29. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Machine learning: proceedings of the 13th international conference, pp 148–156
30. Wilson DR, Martinez TR (1996, June) Heterogeneous radial basis function networks. In: IEEE international conference on neural networks (ICNN'96), pp 1263–1267