A Modified Extreme Learning Machine

Zhixiang Chen [a], Houying Zhu [b] and Yuguang Wang [b,*]

[a] *Department of Mathematics, Shaoxing University, Shaoxing, Zhejiang Province, PR China 312000*

[b] *Department of Information and Mathematics Sciences, China Jiliang University, Hangzhou, Zhejiang Province, PR China 310018*

Abstract

Extreme learning machine (ELM), proposed by Huang et al., has been regarded as a fast learning machine for single hidden layer neural networks. ELM randomly selects the input weights and biases and directly calculates the output weights rather than searching and using iterative methods as traditional algorithms do, which produces the smallest norm and thus obtains the minimum training error. However, some limitations of stochastic choice of hidden layer output matrix will lower to some extend the learning rate and the robustness property of the calculations. This paper proposes a modified ELM algorithm which 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 which improves the speed of training output weights. The experimental results of both regression and classification problems show good performance of our improved ELM algorithm.

Key words: Feedforward neural networks, Back-propagation algorithm, Extreme learning machine, Moore-Penrose generalized inverse.

1 Introduction

Feedforward neural networks have been deeply and systematically studied for their universal approximation capabilities on compact input sets and approximation in a finite set. Cybenko [9] and Funahashi [12] proved that any continuous function can be approximated on a compact set with uniform topology by an sigle-hidden layer feedforward neural network (SLFN) with any continuous

* Corresponding author
Email address: wangyg85@gmail.com (Yuguang Wang).

sigmoidal activation function. Hornik in [14] have shown that any measurable function can be approached with such networks. A variety of results on SLFN approximations to multivariate functions were later established by many authors: [3]-[6], [30]-[31] by Cao, Xu et al., [7],[8] by Chen and Chen, [13] by Hahm and Hong, [22] 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 using classical parametric techniques.

We know that the problems of density and complex in neural network are theoretical bases for algorithms. In practical applications, we often pay close attention to the faster learning algorithms of neural networks in a finite training set.

Huang and Babri [15] showed that a single-hidden layer feedforward neural network with at most N hidden nodes and with almost any nonlinear activation function can exactly learn N distinct samples. Although conventional gradient-based learning algorithms, such as back-propagation (BP) and its variant Levenberg-Marquardt (LM) method, have been extensively used in training of 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 learning algorithms and famous for its good generalization ability. However, fine tuning of SVM kernel parameters is a time-intensive process.

Recently, a new learning algorithm for SLFN named the extreme learning machine (ELM) has been proposed by Huang et al. in [20], [21]. Unlike traditional approaches (such as BP algorithms, SVMs), ELM algorithm has 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 the smallest norm of weights. According to Bartlett (see [2]), feedforward the smaller training error neural networks reach and 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 related problems (see, e.g., [10], [17], [18]-[19], [21]-[22], [28]).

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 $b_i \quad (i = 1, 2, \ldots, N)$.
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 can be used but are not limited to orthogonal projection method, iterative method, and singular value decomposition (SVD). The orthogonal projection, orthogonalization method and iterative method have their limilations since searching and iteration may consume extra training time. The orthogonal project method can be used when the output matrix has full column rank matmix, so the method may not perform well in all applications. The SVD can be generally used to calculate the Moore-Penrose generalized inverse of output matrix in all cases but it costs a lot of time. This paper will design a learning machine which first properly train input weights and biases such that the output matrix is column full-rank, then we can use orthogonal project method to calculate the Moore-Penrose generalized inverse of output matrix.

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

2 Theoretical Deductions of Modified ELM Algorithm with Sigmoidal Activation Function

For N arbitrary distinct sample (X_i, t_i), where $X_i = (x_{i1}, x_{i2}, \cdots, x_{id})^T \in \mathbb{R}^d$ and $t_i = (t_{i1}, t_{i2}, \cdots, t_{im}) \in \mathbb{R}^m$, standard SLFNs with $\tilde{N}$ hidden nodes and activation function $g(x)$ are mathematically modeled as

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

where $\beta_i = (\beta_{i1}, \beta_{i2}, \cdots, \beta_{i\tilde{N}})^T$ is the output weight vector connecting the i-th nodes and output nodes, $W_i = (w_{i1}, w_{i2}, \cdots, w_{id})^T$ is the input weight vector connecting the i-th hidden nodes and the input nodes, and b_i is the threshold of the i-th hidden node. That SLFNs $G_{\tilde{N}}(x)$ can approximate these N samples with zero error means

$$G_{\tilde{N}}(x_j) = t_j, \quad j = 1, 2, \cdots, N,$$

the above N equation can be written as

$$H\beta = T,$$

where

$$H = H(w_1, w_2, \cdots, w_{\tilde{N}};\ b_1, b_2, \cdots, b_{\tilde{N}};\ X_1, X_2, \cdots, X_N)$$
$$= \begin{pmatrix} g(W_1 \cdot X_1 + b_1) & g(W_2 \cdot X_1 + b_2) & \cdots & g(W_{\tilde{N}} \cdot X_1 + b_{\tilde{N}}) \\ g(W_1 \cdot X_2 + b_1) & g(W_2 \cdot X_2 + b_2) & \cdots & g(W_{\tilde{N}} \cdot X_2 + b_{\tilde{N}}) \\ \vdots & \vdots & \cdots & \vdots \\ g(W_1 \cdot X_N + b_1) & g(W_2 \cdot X_N + b_2) & \cdots & g(W_{\tilde{N}} \cdot X_N + b_{\tilde{N}}) \end{pmatrix}_{N \times \tilde{N}},$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_{\tilde{N}} \end{pmatrix}_{\tilde{N} \times m} \quad \text{and} \quad T = \begin{pmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_N \end{pmatrix}_{N \times m}.$$

As named in Huang et al. [15]-[16], 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,$\tilde{N} \ll N$. Then, there may not exist $w_i, b_i, \beta_i (i = 1, 2, \cdots, \tilde{N})$ such that $H\beta = T$. But when H has been given, we can use the least-squares method to find the least-squares solutions of $H\beta = T$. Furthermore, we can reach the special solution $\hat{\beta}$

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

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

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

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

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

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

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*

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

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

$$w_j^1 := \frac{1}{1 + \max\limits_i\{|x_{ij}|\}}, \quad j = 1, 2, \cdots, d,$$

then

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

Let

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

For given $j (1 \le j \le d)$, if $y_{ij}^1 = 0$ for all $i = 1, 2, \cdots, n-1$, then let $n_j = 2d$; if not

$$n_j = \frac{4d}{\min\limits_{y_{ij}^1 \neq 0}\{y_{ij}^1\}}.$$

Obviously, we have

$$n_j \ge 2d, \quad j = 1, 2, \cdots, d.$$

Define

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

and

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

For fixed $i (1 \le i \le n-1)$, from $X_i \prec X_{i+1}$ we see that there exists $k_0 \in \mathbb{N}$ such that

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

Write

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

Since

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

And

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

We get $W \cdot X_{i+1} - W \cdot X_i > 0$, that is, $W \cdot X_i < W \cdot X_{i+1}$. This finishes the proof of Lemma 2.1. $\square$

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

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

Set

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

we have

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

Let $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ is a set of samples, and $x_1 < x_2 < \cdots < x_n$. Then, we have

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

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

is nonsingular.

Proof. Since

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

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

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

and

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

We choose

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

it follows that

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

In turn, subtract the 2th row from the first row, the 3th row from the 2th row, $\cdots\cdots$, and the 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 i-th row can be estimated as follows. For $j=1,\ldots,i-1$, since

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

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

For $j=i$,

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

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

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

implies

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

For $j = n$, we can similarly obtain that

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

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

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

which implies

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

and

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

Therefore,

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

that is, $\widetilde{G}_\sigma$ is strictly diagonally dominant which guarantees that $\widetilde{G}_\sigma$ is non-singular and thus the non-singularity of G_σ. $\square$

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

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

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

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

where A satisfies $\delta_\sigma(A) < 1/(4N)$. From Lemma 2.2, we have

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

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

is nonsingular.

By the knowledge of matrix and Theorem 2.1, we have

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

$$H = \begin{pmatrix} \sigma(W_1 \cdot X_1 + b_1) & \sigma(W_2 \cdot X_1 + b_2) & \cdots & \sigma(W_{\tilde{N}} \cdot X_1 + b_{\tilde{N}}) \\ \sigma(W_1 \cdot X_2 + b_1) & \sigma(W_2 \cdot X_2 + b_2) & \cdots & \sigma(W_{\tilde{N}} \cdot X_2 + b_{\tilde{N}}) \\ \vdots & \vdots & \cdots & \vdots \\ \sigma(W_1 \cdot X_N + b_1) & \sigma(W_2 \cdot X_N + b_2) & \cdots & \sigma(W_{\tilde{N}} \cdot X_N + b_{\tilde{N}}) \end{pmatrix}_{N \times \tilde{N}}$$

is full column rank.

3 An Improved ELM Using Sigmoidal Activation Function

According to the discussion of Sectioin 2, we can based on ELM algorithm propose a new algorithm which uses Sigmoidal function as its activation function and can make use of orthogonal projection method to calculate the output weights. We states the modified extreme learning machine 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 bias 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.

$$w_j^1 = \frac{1}{\max\limits_{i=1,2,\ldots,N}\{|x_{ij}|\}} > 0,$$

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

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

$$n_j = \begin{cases} 2d, & \text{if } y_{ij}^1 = 0 \text{ for all } i = 1, 2, \ldots, N-1, \\ \dfrac{4d}{\min\limits_{y_{ij}^1 \neq 0}\left\{y_{ij}^1\right\}}, & \text{else,} \end{cases}$$

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

Set

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

Let $a_i = W \cdot X_i$,

$$\begin{aligned} W_i &= \frac{2A}{a_i - a_i} W, \quad i = 1, 2, \ldots, N-1, \\ W_N &= W_{N-1}, \\ b_i &= A + \frac{2Aa_i}{a_{i+1} - a_i}, \quad i = 1, 2, \ldots, N = 1, \\ b_N &= A + \frac{2Aa_N}{a_N - a_{N-1}}. \end{aligned}$$

Step 2: Calculate output weights $\beta = (\beta_1, \beta_2, \ldots, \beta_N)$ $(i = 1, 2, \ldots, N)$.

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

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

Then

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

The ELM proved 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 [28], 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 largely 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 non-singularity of $\mathbf{H}^T\mathbf{H}$.

4 Performance Verification

In this section, the performance of the proposed new ELM learning algorithm is measured compared with the ELM 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 $g(x) = 1/(1 + e^{-x})$.

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 ELM and the modified ELM algorithms. The target function is as follows.

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

A training set $\{(X_i, t_i)\}$ and testing set $\{(X_i, t_i)\}$ with 5000 samples are respectively created, where X_i in both training and testing data is distributed in $[-10, 10]$ with uniform step length. 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.

Algorithms	Time		Accuracy		#Nodes
	Training	Testing	Training	Testing	
ELM	0.0684	0.0231	0.1144	0.0061	20
			$\pm 2.1315 \times 10^{-5}$	$\pm 3.9747 \times 10^{-4}$	
modified ELM	0.0563	0.0197	0.1243	0.0.0466	20
			$\pm 1.4019 \times 10^{-16}$	$\pm 5.6075 \times 10^{-17}$	20

Table 1
Performance comparison for learning function: SinC

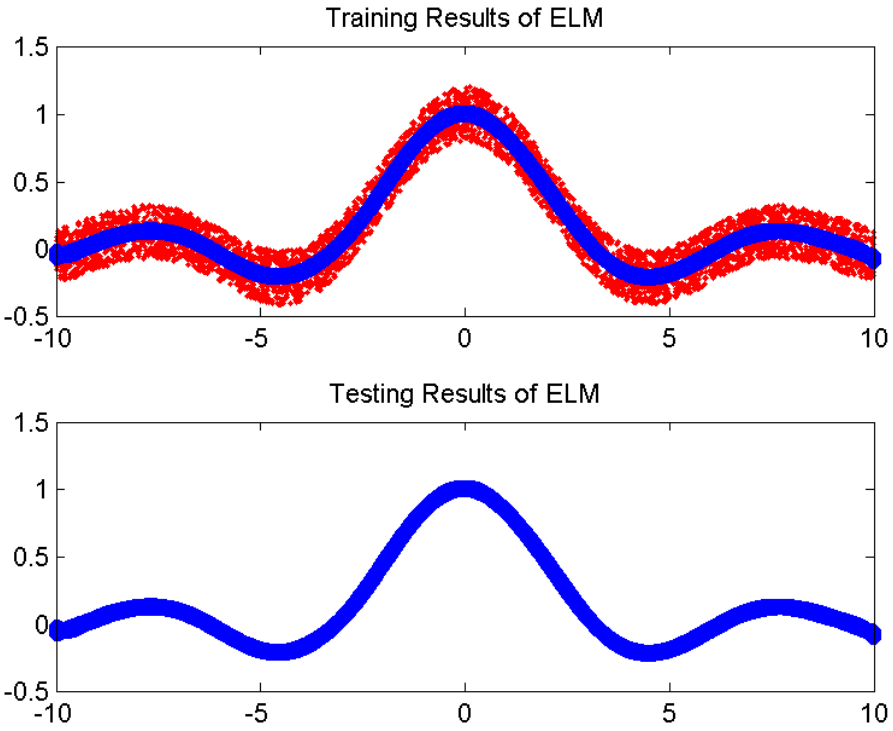

Figure 1. Outputs of ELM learning algorithm

It can be seen from Table 1 that the modified ELM algorithm spent $0.0563s$, $0.0197s$ CPU time training and testing respectively whereas it takes $0.0684s$ and 0.0231 for the ELM algorithm to complete the same process. Therefore the new algorithm is quicker than ELM. Simultaneously, for the accuracy of training and testing of modified ELM, the standard deviation of training and testing time are smaller than those of ELM, so the new algorithm is more stable.

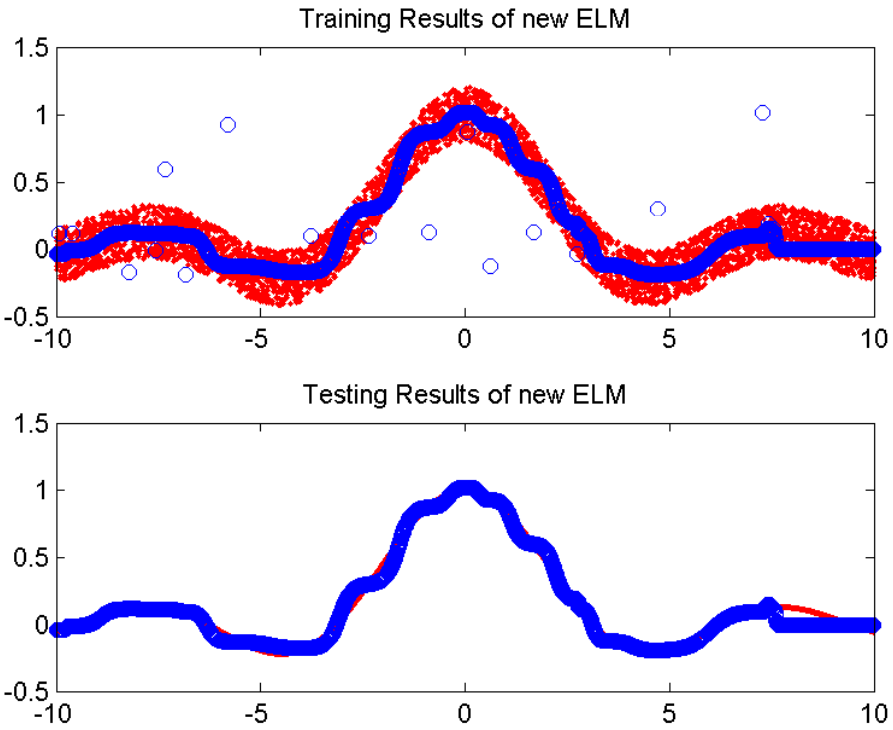

Figure 2. Outputs of modified ELM learning algorithm

Figure 2 shows the expected and approximated results of modified ELM algorithm and Figure 1 shows the true and the approximated results of ELM algorithm. The results show that the modified ELM algorithm has a good performance as ELM does.

4.2 Benchmarking with Practical problems applications

Performance comparison of the proposed modified ELM and the ELM algorithms for four real problems are carried out in this section. Classification and regression tasks are included in four real problems. Two classification tasks including Diabetes, Glass Identification (Glass ID), and two regression tasks including Housing and Slump (Concrete Slump). All the data sets are from UCI repository of machine learning databases [1]. The speculation of each database is shown in the Table 2. For the databases that have only one data table, as conducted in [11], [25], [26], [29], 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, we conduct fifty trials

for the two algorithms,and then take the average of fifty times as final results. The results are reported in Table 3, Table 4 and Table 5, which show that in our simulation, on the average, ELM and the modified ELM algorithms are discriminating sightly in the learning time which is reported in Table 5, they both have a fast learning rate.However, contrasting with ELM,the modified ELM has a more stabilized accuracy of training and testing which can refer to the Table 4, especially in the case regression problems, this advantage is obvious. From Table 3, we may observe that the modified ELM algorithm is better than ELM in the accuracy of learning for the regression cases.

Data sets	# Observations		# Attributes	Associated Tasks	#Nodes
	Training	Testing	Continuous		
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

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

Data sets	ELM		new ELM	
	Training	Testing	Training	Testing
Diabetes	0.7892	0.7770	0.6901	0.6581
Glass ID	0.9485	0.526	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 3
Comparison training and testing accuracy (error) of ELM and modified ELM

Compared to ELM algorithm, the modified ELM selects the input weights and biases, which helps to avoid risk of the random errors as we can see from the Figure 3 - Figure 18.

Data sets	ELM		modified ELM	
	Training	Testing	Training	Testing
Diabetes	0.0116	0.0275	0.0142	0.0281
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^{8}	5.3832×10^{-15}	1.2561×10^{-14}

Table 4
Comparison of training and testing RMSE of ELM and modified ELM

Data sets	ELM(s)		modified ELM(s)	
	Training	Testing	Training	Testing
Diabetes	0.0034	0.0028	0.0078	0.0044
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

Table 5
Comparison of average training and testing time of ELM and modified ELM

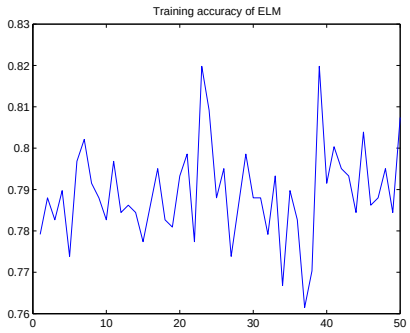

Figure 3. Training accuracy of ELM for Diabetes

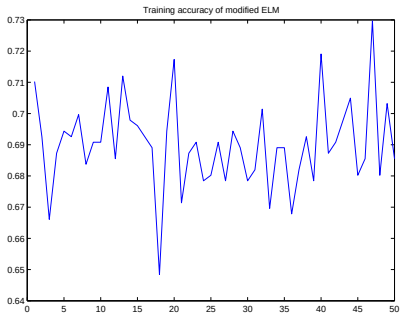

Figure 4. Training accuracy of modified ELM for Diabetes

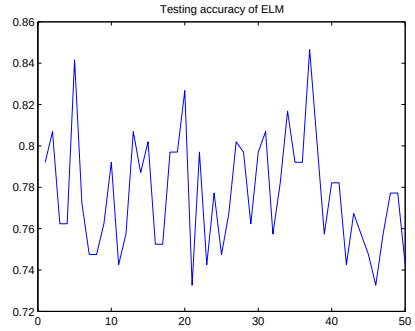

Figure 5. Testing accuracy of ELM for Diabetes

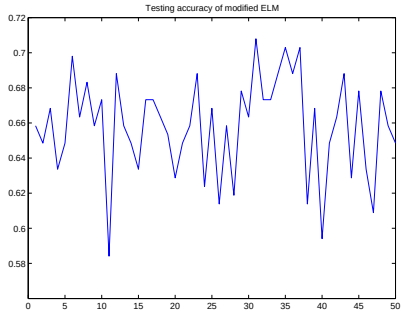

Figure 6. Testing accuracy of modified ELM for Diabetes

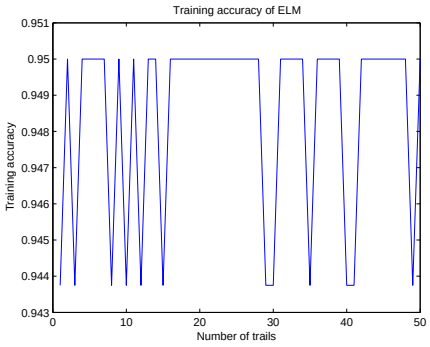

Figure 7. Training accuracy of ELM for GlassID

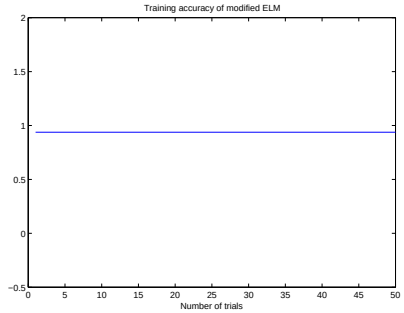

Figure 8. Training accuracy of modified ELM for GlassID

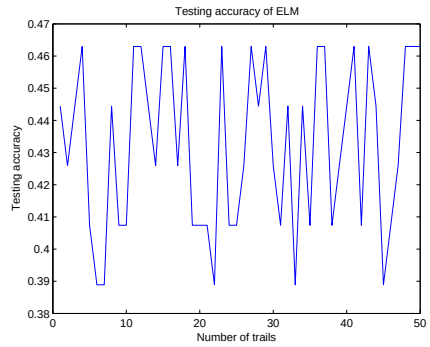

Figure 9. Testing accuracy of ELM for GlassID

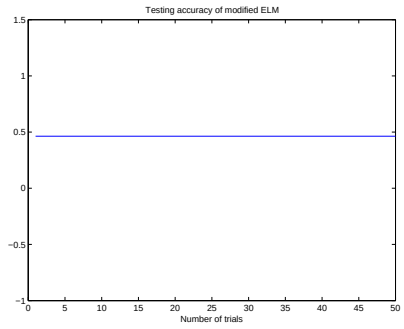

Figure 10. Testing accuracy of modified ELM for GlassID

5 Conclusions

This paper proposed a modified ELM algorithm based on the ELM for training single-hidden layer feedforward neural networks (SLFNs) in an attempt to solve the least-squares minimization of SLFNs in a more effective way and meanwhile solves the open problem of [28]. Compared with ELM, the modified ELM algorithm proposed have several features as follows.

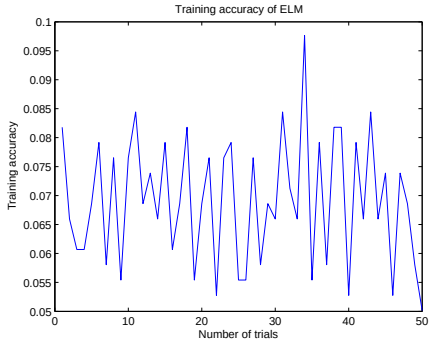

Figure 11. Training accuracy of ELM for Housing

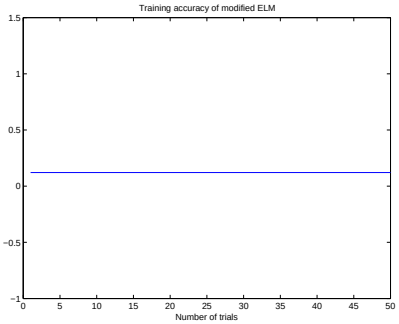

Figure 12. Training accuracy of modified ELM for Housing

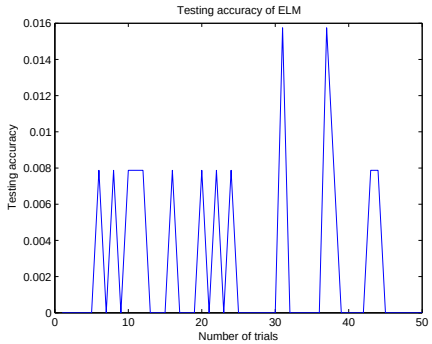

Figure 13. Testing accuracy of ELM for Housing

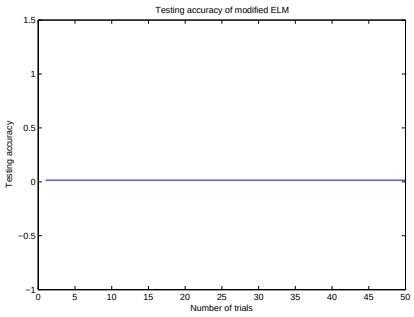

Figure 14. Testing accuracy of modified ELM for Housing

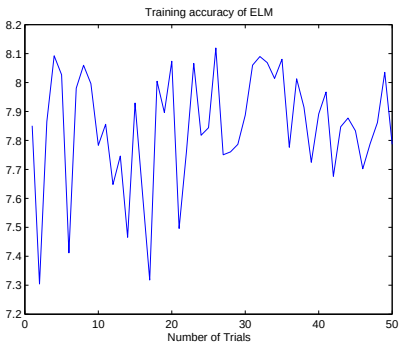

Figure 15. Training accuracy of ELM for Slump

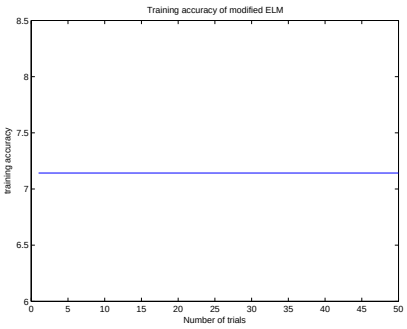

Figure 16. Training accuracy of modified ELM for Slump

(1) The learning speed of modified ELM is as fast as ELM. The main difference between the modified ELM and ELM algorithms lie in the selection of input weights and biases. Our modified algorithm selects the input weights and biases properly, which, however, consumes little time compared with the training time of output weights.

(2) The proposed ELM by making proper selection of input weights and biases of the neural networks avoids the risk of yielding singular or

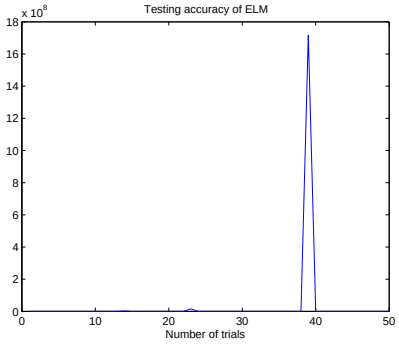

Figure 17. Testing accuracy of ELM for Slump

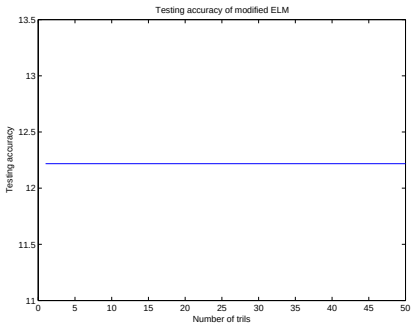

Figure 18. Testing accuracy of modified ELM for Slump

not full column rank hidden layer output matrix $\mathbf{H}$. This ensures that the orthogonal projection which is a fast method to calculate can be used to calculate the Moore-Penrose generalized inverse of $\mathbf{H}$.

(3) The modified ELM algorithm overcome the shortcomings of EELM, the algorithm proposed in [28], that is, it can use Sigmoidal function as activation function to train the networks but still keep the qualities of ELM and EELM: fast and stable and in many cases perform better.

(4) It is worthwhile to point out that our modified ELM algorithm performs very well in most cases listed above regardless of the number of samples, which is another beautiful feature of the new ELM and to some extent superior to EELM.

(5) Another point that should be mentioned is that in our algorithm in order to sort the samples by affine transformation $X \mapsto W \cdot X + b$, we adopt the similar method of decimal numeral system as in [28]. The problem of "curse of dimensionality" still exists here and when encountering high-dimensional data, our algorithm become less effective sometimes.

References

[1] C. Blake, C. Merz, "UCI repository of machine learning databases", *http://www.ics.uci.edu/˜mlearn/MLRepository.html, Department of Information and Computer Sciences, University of California, Irvine*, USA, 1998.

[2] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, pp. 525–536, 1998.

[3] F. L. Cao, T. F. Xie, Z. B. Xu, "The estimate for approximation error of neural networks: a constructive approach," *Neurocomputing*, vol. 71, pp. 626–630, 2008.

[4] F. L. Cao, Y. Q. Zhang, Z. R. He, "Interpolation and rate of convergence by a class of neural networks," *Applied Mathematical Modelling*, vol. 33, pp. 1441–1456, 2009.

[5] F. L. Cao, R. Zhang, "The Errors of Approximation for Feedforward Neural Networks in the Lp Metric," *Mathematical and Computer Modelling*, vol. 49, pp. 1563–1572, 2009.

[6] F. L. Cao, S. B. Lin, Z. B. Xu, "Approximation capabilities of interpolation neural networks," *Neurocomputing*, vol. 74, pp. 457–460, 2010.

[7] 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*, vol. 6, pp. 904–910, 1995.

[8] 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*, vol. 6, pp. 911–917, 1995.

[9] G. Cybenko, "Approximation by superposition of sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[10] 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*, vol. 20, pp. 1352–1357, 2009.

[11] Y. Freund, R.E. Schapire, "Experiments with a new boosting algorithm," *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148–156, 1996.

[12] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[13] N. Hahm, B. I. Hong, "An approximation by neural networks with a fixed weight," *Computers & Mathematics with Applications*, vol. 47, pp. 1897-1903, 2004.

[14] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[15] G. B. Huang, H. A. Babri, "Upper bounds on the number of hidden neurons in ffeedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, pp. 224–229, 1998.

[16] G. B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol. 14, pp. 274–281, 2003.

[17] G. B. Huang, L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, pp. 3056–3062, 2007.

[18] G. B. Huang, L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, pp. 3460–3468, 2008.

[19] G. B. Huang, L. Chen, C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, pp. 879–892, 2006.

[20] G. B. Huang, Q. Y. Zhu, C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[21] G. B. Huang, Q. Y. Zhu, C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," *Proceedings of 2004 IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, July 2004.

[22] Y. Lan, Y. C. Soh, G. B. Huang, "Random search enhancement of error minimized extreme learning machine," *European Symposium on Artificial Neural Networks*, pp. 327–332, April 2010.

[23] M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, pp. 861–867, 1993.

[24] C. R. Rao, S. K. Mitra, "Generalized inverse of matrices and its applications," *Wiley*, New York, 1971.

[25] G. Rätsch, T. Onoda, K.R. Müller, "An improvement of AdaBoost to avoid overfitting," *Proceedings of the Fifth International Conference on Neural Information Processing (ICONIP' 1998)*, 1998.

[26] E. Romero, R. Alquézar, "A new incremental method for function approximation using feed-forward neural networks," *Proceedings of INNS-IEEE International Joint Conference on Neural Networks (IJCNN2002)*, pp. 1968–1973, 2002.

[27] D. Serre, "Matrices: theory and applications," *Springer*, New York, 2000.

[28] Y. G. Wang, F. L. Cao, Y. B. Yuan, "A study on effectiveness of extreme learning machine," *Neurocomputing*, in press, 2011.

[29] D. R. Wilson, T. R. Martinez, "Heterogeneous radial basis function networks," *Proceedings of the International Conference on Neural Networks (ICNN 96)*, pp. 1263–1267, June 1996.

[30] Z. B. Xu, F. L. Cao, "The essential order of approximation for neural networks," *Science in China (F)*, vol. 47, pp. 97–112, 2004.

[31] Z. B. Xu, F. L. Cao, "Simultaneous L^p approximation order for neural networks," *Neural Networks*, vol. 18, pp. 914–923, 2005.