# Enhanced Random Search Based Incremental Extreme Learning Machine

Guang-Bin Huang [a,*] and Lei Chen [a,b]

[a]*School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798*

[b]*School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543*

**Abstract**

Recently an incremental algorithm referred to as I-ELM was proposed by Huang, et al [1], which randomly generates hidden nodes and then analytically determine the output weights. Huang, et al [1] has proved in theory that although additive or RBF hidden nodes are generated randomly the network constructed by I-ELM can work as a universal approximator. During our recent study, it is found that some of hidden nodes in such networks may play very minor contribution to the network output and thus may eventually increase the network complexity. In order to avoid this issue and to obtain a more compact network architecture, this paper proposes an enhanced method for I-ELM (referred to as EI-ELM). At each learning step, several hidden nodes are randomly generated and among them the hidden node leading to the largest residual error decreasing will be added to the existing network and the output weight of the network will be calculated in a same simple way as in the original I-ELM. Generally speaking, the proposed enhanced I-ELM works for the widespread type of piecewise continuous hidden nodes.

*Key words:* Incremental extreme learning machine, convergence rate, random hidden nodes, random search.

## 1 Introduction

In the past decades **s**ingle-hidden-**l**ayer **f**eedforward **n**etworks (SLFNs) (may or may not be *neural* based) have been investigated extensively from both

* Corresponding author
*Email address:* egbhuang@ntu.edu.sg (Guang-Bin Huang).

theoretical and application aspects. SLFN network functions with $n$ hidden nodes can be represented by

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} \beta_i g_i(\mathbf{x}) = \sum_{i=1}^{n} \beta_i G(\mathbf{x}, \mathbf{a}_i, b_i), \mathbf{a}_i \in \mathbf{C}^d, \mathbf{x} \in \mathbf{C}^d, b_i \in C, \beta_i \in C \quad (1)$$

where $g_i$ or $G(\mathbf{x}, \mathbf{a}_i, b_i)$ denotes the output function of the $i$th hidden node and $\beta_i$ is the (output) weight of the connection between the $i$th hidden node and the output node. The hidden nodes of SLFNs can be neuron alike or non-neuron alike, including additive or RBF (radial basis function) type of nodes [1], multiplicative nodes, fuzzy rules [2], fully complex nodes [3,4], hinging functions [5], high-order nodes [6], ridge polynomials [7], wavelets [8,9], and Fourier terms [10], etc.

For the specific example of the standard single-hidden-layer feedforward *neural* networks, three types of hidden nodes have been mainly used:

1) Additive hidden nodes:

$$g_i(\mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \mathbf{a}_i \in \mathbf{R}^d, b_i \in R \quad (2)$$

where $g$ is the activation function of hidden nodes.

2) RBF hidden nodes:

$$g_i(\mathbf{x}) = g\left(b_i \|\mathbf{x} - \mathbf{a}_i\|\right), \mathbf{a}_i \in \mathbf{R}^d, b_i \in R \quad (3)$$

3) Fully complex hidden nodes[3]:

$$g_i(\mathbf{x}) = \prod_{l=1}^{s_i} g\left(\mathbf{a}_{il} \cdot \mathbf{x} + b_i\right), \mathbf{a}_{il} \in \mathbf{C}^d, \mathbf{x} \in \mathbf{C}^d, b_i \in C, \beta_i \in C \quad (4)$$

where $s_i$ is an integer constant.

However, the output functions $G(\mathbf{x}, \mathbf{a}_i, b_i)$ to be studied in this paper can be non-neuron alike and can be different from the abovementioned as long as they are nonlinear and piecewise continuous.

Conventional neural network theories [11,12] show that SLFNs with additive or RBF hidden nodes can work as universal approximators when all the parameters of the networks are allowed adjustable. However, tuning all the parameters of the networks may cause learning complicated and inefficient since tuning based learning may easily converge to local minima and/or may generally be very slow due to improper learning steps. Many iterative learning steps may be required by such learning algorithms in order to obtain better learning performance. It may be difficult to apply tuning methods in neural networks with non-differentiable activation functions such as threshold networks. Interestingly, different from the common understanding on the neural networks White [13] found that "random search" over input to hidden layer connections is computational efficient in SLFNs with affine transformation and no additional learning is required. Unlike the conventional neural network theories, Huang, et al [1] has recently proved that SLFNs with additive or RBF hidden nodes and with randomly generated hidden node parameters $(\mathbf{a}_i, b_i)$ [1] can work as universal approximators by only calculating the output weights $\beta_i$ linking the hidden layer to the output nodes. In such SLFNs implementations, the activation functions for additive nodes can be any bounded nonconstant piecewise continuous functions and the activation functions for RBF nodes can be any integrable piecewise continuous functions. The resulting method referred to as Incremental Extreme Learning Machines (I-ELM) [1] randomly adds nodes to the hidden layer one by one and freezes the output weights of the existing hidden nodes when a new hidden node is added. The original batch learning mode Extreme Learning Machines (ELM) [4,14–17] and their online sequential learning mode ELM (OS-ELM) [2,18] are developed for SLFNs with fixed network architectures. The activation functions used in ELM include differentiable [14,15,17] and nondifferentiable [16] functions, continuous [14,15,17] and noncontinuous [16] functions, etc.

I-ELM [1] randomly adds nodes to the hidden layer one by one and freezes the output weights of the existing hidden nodes when a new hidden node is added. I-ELM [1] is a "*random search*" method in the sense that in theory the residual error of I-ELM will decrease and I-ELM moves towards the target function further whenever a hidden node is randomly added. Huang, et al [19] also shows its universal approximation capability for the case of fully complex hidden nodes. Huang, et al [20] recently proposed a convex optimization method based incremental method to further improve the convergence rate of I-ELM by allowing properly adjusting the output weights of the existing hidden nodes when a new hidden node is added. Huang, et al [20] also extends the ELM from additive and RBF type of SLFNs to "generalized" SLFNs (including those which are usually not considered as the standard single-hidden layer feedforward *neural* networks.) where any type of piecewise continuous hidden nodes $G(\mathbf{x}, \mathbf{a}_i, b_i)$ could be used. These hidden nodes could include additive or RBF (radial basis function) type of nodes, multiplicative nodes, fuzzy rules, fully complex nodes, hinging functions, high-order nodes, ridge polynomials, wavelets, and Fourier terms, etc.

In this paper we propose a simple improved implementation of I-ELM in order to achieve a more compact network architecture than the original I-ELM. Similar to the original I-ELM, the improved I-ELM is fully automatic in the sense that except for target errors and the allowed maximum number of hidden

[1] In theory [1], from the function approximation point of view the hidden node parameters of the single-hidden layer feedforward neural networks are independent of the training data and of each other.

nodes no control parameters need to be manually set by users. Different from the original I-ELM which is studied in the real domain, this paper simply discusses the proposed enhanced I-ELM in the complex domain which considers the real domain as its specific case.

## 2 Proposed Enhanced I-ELM

Let $L^2(X)$ be a space of functions $f$ in a measurable compact subset $X$ of the $d$-dimensional space $\mathbf{C}^d$ such that $|f|^2$ are integrable. For $u, v \in L^2(X)$, the inner product $\langle u, v \rangle$ is defined by $\langle u, v \rangle = \int_X u(\mathbf{x})\overline{v(\mathbf{x})}d\mathbf{x}$. The closeness between network function $f_n$ and the target function $f$ is measured by the $L^2$ distance:

$$\|f_n - f\| = \left[\int_X (f_n(\mathbf{x}) - f(\mathbf{x}))\,\overline{(f_n(\mathbf{x}) - f(\mathbf{x}))}d\mathbf{x}\right]^{1/2} \tag{5}$$

In this paper, the sample input space $X$ is always considered as a bounded measurable compact subset of the space $\mathbf{C}^d$.

The I-ELM [1] adds random nodes to the hidden layer one by one and freezes the output weights of the existing hidden nodes after a new hidden node is added. Denote the residual error of $f_n$ as $e_n \equiv f - f_n$ where $f \in L^2(X)$ is the target function. Huang, et al[1] has proved (for $X \in \mathbf{R}^d$) that:
**Lemma 2.1.** *[1] Given any bounded nonconstant piecewise continuous function $g : \mathbf{R} \to \mathbf{R}$ for additive nodes or any integrable piecewise continuous function $g : \mathbf{R} \to \mathbf{R}$ and $\int_{\mathbf{R}} g(x)dx \neq 0$ for RBF nodes, for any continuous target function $f$ and any randomly generated function sequence $\{g_n\}$, $\lim_{n\to\infty} \|f - f_n\| = 0$ holds with probability one if*

$$\beta_n = \frac{\langle e_{n-1}, g_n \rangle}{\|g_n\|^2} \tag{6}$$

Lemma 2.1 shows that different from the common understanding in neural networks, from the function approximation point of view the hidden nodes of SLFNs need not be tuned at all. In order to train SLFNs with additive or RBF nodes one only need to randomly assign the hidden nodes ("*random searching*") and analytically calculate the output weights only. No tuning is required in the training of SLFNs. Furthermore, the universal approximation capability of ELM for fixed network architectures [14–17] is the straightforward corollary of Lemma 2.1. In fact, seen from the proof of [1] Lemma 2.1 is valid if and only if $G(\mathbf{x}, \mathbf{a}, b)$ is piecewise continuous and $span\{G(\mathbf{x}, \mathbf{a}, b) : (\mathbf{a}, b) \in \mathbf{C}^d \times C\}$ is dense in $L^2$. Based on the same proof of [1] we can further extends [1] from the standard single-hidden layer feedforward neural networks (with additive or

RBF hidden nodes) to "generalized" SLFNs $f_n(\mathbf{x}) = \sum_{i=1}^{n} \beta_i G(\mathbf{x}, \mathbf{a}_i, b_i)$ with any type of hidden nodes $G(\mathbf{x}, \mathbf{a}_i, b_i)$.

**Theorem 2.1.** *Given a SLFN with any nonconstant piecewise continuous hidden nodes $G(\mathbf{x}, \mathbf{a}, b)$, if span$\{G(\mathbf{x}, \mathbf{a}, b) : (\mathbf{a}, b) \in \mathbf{C}^d \times C\}$ is dense in $L^2$, then for any continuous target function $f$ and any function sequence $\{g_n(\mathbf{x}) = G(\mathbf{x}, \mathbf{a}_n, b_n)\}$ randomly generated based on any continuous sampling distribution,* $\lim_{n\to\infty} \|f - (f_{n-1} + \beta_n g_n)\| = 0$ *holds with probability one if*

$$\beta_n = \frac{\langle e_{n-1}, g_n \rangle}{\|g_n\|^2} \tag{7}$$

*Proof.* The proof method of I-ELM [1] can be adopted to prove the validity of this theorem without any major modification.

Similar to [1], for any function sequence $\{g_n(\mathbf{x}) = G(\mathbf{x}, \mathbf{a}_n, b_n)\}$ randomly generated based on any continuous sampling distribution we can prove that $\|e_n\| = \|f - (f_{n-1} + \beta_n g_n)\|$ achieves its minimum if

$$\beta_n = \frac{\langle e_{n-1}, g_n \rangle}{\|g_n\|^2} \tag{8}$$

Furthermore, when $\beta_n = \frac{\langle e_{n-1}, g_n \rangle}{\|g_n\|^2}$, the sequence $\{\|e_n\|\}$ decreases and converges.

As *span*$\{G(\mathbf{x}, \mathbf{a}, b)\}$ is dense in $L^2$ and $G(\mathbf{x}, \mathbf{a}, b)$ is a nonconstant piecewise continuous function, in order to prove $\lim_{n\to+\infty} \|e_n\| = 0$, seen from the proof of the original I-ELM [1] we only need to prove $e_n \perp (e_{n-1} - e_n)$.

According to formula (1), we have

$$e_n = f - f_n = e_{n-1} - \beta_n g_n \tag{9}$$

Furthermore, we have

$$\langle e_n, g_n \rangle = \langle e_{n-1} - \beta_n g_n, g_n \rangle = \langle e_{n-1}, g_n \rangle - \beta_n \|g_n\|^2 \tag{10}$$

According to formula (8), we further have

$$\langle e_n, g_n \rangle = \langle e_{n-1}, g_n \rangle - \frac{\langle e_{n-1}, g_n \rangle}{\|g_n\|^2} \cdot \|g_n\|^2 = \langle e_{n-1}, g_n \rangle - \langle e_{n-1}, g_n \rangle = 0 \tag{11}$$

Thus, from formula (9) and (11) we have $\langle e_n, e_n - e_{n-1} \rangle = \langle e_n, -\beta_n g_n \rangle = 0$, which means $e_n \perp (e_n - e_{n-1})$.

This completes the proof of this theorem. □

Different from the convex optimization based incremental extreme learning machine (CI-ELM)[20], Theorem 2.1 is valid for the case where all the output weights of the existing hidden nodes are frozen when a new hidden node is added.

It should be noted that although hidden nodes can be added randomly some newly added hidden nodes may make residual error reduce less and some newly added hidden nodes may make residual error reduce more. Based on this observation, in this paper we propose an enhanced I-ELM algorithm. In this new method, at any step, among $k$ trial of hidden nodes, the hidden nodes with greatest residual error reduction will be added, where $k$ is a fixed constant.

**Theorem 2.2.** *Given a SLFN with any nonconstant piecewise continuous hidden nodes $G(\mathbf{x}, \mathbf{a}, b)$, if span$\{G(\mathbf{x}, \mathbf{a}, b) : (\mathbf{a}, b) \in \mathbf{C}^d \times C\}$ is dense in $L^2$, for any continuous target function $f$ and any randomly generated function sequence $\{g_n\}$ and any positive integer $k$, $\lim_{n\to\infty} \|f - f_n^*\| = 0$ holds with probability one if*

$$\beta_n^* = \frac{\left\langle e_{n-1}^*, g_n^* \right\rangle}{\|g_n^*\|^2} \tag{12}$$

*where $f_n^* = \sum_{i=1}^{n} \beta_i^* g_i^*$, $e_n^* = f - f_n^*$ and $g_n^* = \{g_i | \min_{(n-1)k+1\le i\le nk} \|(f - f_{n-1}^*) - \beta_n g_i\|\}$.*

*Proof.* Given the function sequence $\{g_n\}$ which is randomly generated based on any continuous sampling distribution probability, for fixed $k$ we can choose an element of the $n$th segment $\{g_{(n-1)k+1}, \cdots, g_{nk}\}$ as $g_n^*$ which minimizes $\|(f - f_{n-1}^*) - \beta_n g_i\|$, $i = (n-1)k + 1, \cdots, nk$.

Similar to [1], we can prove that $\|e_n^*\| = \|f - (f_{n-1}^* + \beta_n^* g_n^*)\|$ achieves its minimum, and the sequence $\{\|e_n\|\}$ decreases and converges if $\beta_n^* = \frac{\langle e_{n-1}^*, g_n^* \rangle}{\|g_n^*\|^2}$.

As span$\{G(\mathbf{x}, \mathbf{a}, b)\}$ is dense in $L^2$ and $G(\mathbf{x}, \mathbf{a}, b)$ is a nonconstant piecewise continuous function, seen from the proof of the original I-ELM [1] (Lemma II.5 and Lemma II.6 of [1] are also valid for the sequence $\{g_n^*\}_{n=1}^{\infty}$) in order to prove $\lim_{n\to+\infty} \|e_n^*\| = 0$ we only need to prove $e_n^* \perp (e_{n-1}^* - e_n^*)$.

As $e_n^* = f - f_n^* = e_{n-1}^* - \beta_n^* g_n^*$, we have

$$\langle e_n^*, g_n^* \rangle = \left\langle e_{n-1}^* - \beta_n^* g_n^*, g_n^* \right\rangle = \left\langle e_{n-1}^*, g_n^* \right\rangle - \beta_n^* \|g_n^*\|^2 \tag{13}$$

According to formula (8), we further have

$$\langle e_n^*, g_n^* \rangle = \left\langle e_{n-1}^*, g_n^* \right\rangle - \frac{\left\langle e_{n-1}^*, g_n^* \right\rangle}{\|g_n^*\|^2} \cdot \|g_n^*\|^2 = \left\langle e_{n-1}^*, g_n^* \right\rangle - \left\langle e_{n-1}^*, g_n^* \right\rangle = 0 \tag{14}$$

Thus, we have $\left\langle e_n^*, e_n^* - e_{n-1}^* \right\rangle = \langle e_n^*, -\beta_n^* g_n^* \rangle = 0$, which means $e_n^* \perp (e_n^* - e_{n-1}^*)$.

This completes the proof of this theorem. $\square$

According to Theorem 2.2, when the $n$th hidden node is added, the weight $\beta_n$ linking the new node to the output node should be chosen as $\frac{\langle e_{n-1}^*, g_n^* \rangle}{\|g_n^*\|^2}$. In practice, it could not be calculated since the exact functional form of $e_{n-1}^*$ is unavailable. Similar to I-ELM [1], a consistent estimate of the weight $\beta_n$ based on the training set is (originally proposed by Kwok and Yeung[21] for the case where the hidden nodes are not *randomly* added):

$$\beta_n = \frac{E \cdot H^T}{H \cdot H^T} = \frac{\sum_{p=1}^{N} e(p)h(p)}{\sum_{p=1}^{N} h^2(p)} \tag{15}$$

where $h(p)$ is the activation of the new hidden node for the input of $p$th training sample and $e(p)$ is the corresponding residual error before this new hidden neuron is added. $H = [h(1), \cdots, h(N)]^T$ is the activation vector of the new node for all the $N$ training samples and $E = [e(1), \cdots, e(N)]^T$ is the residual vector before this new hidden node added. In real applications, one may not really wish to get zero approximation error by adding infinite neurons to the network, a maximum number of hidden neurons is normally given. Thus, such an incremental constructive method[2] [3] for SLFNs can be summarized as follows:

**EI-ELM Algorithm:** Given a training set $\aleph = \{(\mathbf{x}_i, t_i) | \mathbf{x}_i \in \mathbf{C}^d, t_i \in C, i = 1, \cdots, N\}$, hidden node output function $G(\mathbf{x}, \mathbf{a}, b)$, maximum number $L_{\max}$ of

[2] White, et al in their seminal works [13,22–24] suggested to use "random hidden nodes" in the SLFNs augmented by connections from the input layer to output layer): $f_n(\mathbf{x}) = \theta \cdot \mathbf{x} + \sum_{i=1}^{n} \beta_i g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$, especially White [24] proposed an interesting incremental learning algorithm named QuickNet. Similar to I-ELM and EI-ELM, QuickNet randomly adds hidden nodes one by one. However, unlike I-ELM and EI-ELM which only simply calculates the output weight of the newly added hidden node, QuickNet readjusts the output weights of the existing hidden nodes after a new hidden node is added which may require more computational burden than I-ELM and EI-ELM. Simply speaking, EI-ELM can be considered a further simplification of White's incremental methods[13,24].

[3] Instead of randomly adding hidden nodes, Kwok and Yeung[21] adds the optimal hidden nodes one by one. Unlike White's works[13,22–24] and I-ELM and EI-ELM, optimization learning is required in Kwok and Yeung[21]. When a new hidden node is added both Kwok and Yeung[21] and EI-ELM do not readjust the existing hidden nodes and they compute the output weights in a similar way: $\beta_n^* = \frac{\langle e_{n-1}^*, g_n^* \rangle}{\|g_n^*\|^2}$. The newly added node $g_n^*$ in EI-ELM will be as close to the optimal hidden node in Kwok and Yeung[21] as required if the number of selecting trial $k$ in EI-ELM is sufficiently large. In this sense EI-ELM has unified the I-ELM, Kwok and Yeung[21] and White's work[13,24].

hidden nodes, maximum number $k$ of trials of assigning random hidden nodes at each step, and expected learning accuracy $\epsilon$,

```
step 1 Initialization: Let $L = 0$ and residual error $E = t$, where $t = [t_1, \cdots, t_N]^T$.
step 2 Learning step:
       while $L < L_{\max}$ and $\|E\| > \epsilon$
       (a) Increase by 1 the number of hidden nodes $L$: $L = L + 1$.
       (b) for $i = 1 : k$
           (i) Assign random parameters $(\mathbf{a}_{(i)}, b_{(i)})$ for the new hidden node
               $L$ according to any continuous sampling distribution probability.
           (ii) Calculate the output weight $\beta_{(i)}$ for the new hidden node:
                $$\beta_{(i)} = \frac{E \cdot H_{(i)}^T}{H_{(i)} \cdot H_{(i)}^T} \qquad (16)$$
           (iii) Calculate the residual error after adding the new hidden node $L$:
                $$E_{(i)} = E - \beta_{(i)} \cdot H_{(i)} \qquad (17)$$
           endfor
       (c) Let $i^* = \{i | \min_{1 \leq i \leq k} \|E_{(i)}\|\}$. Set $E = E_{(i)}$, $\mathbf{a}_L = \mathbf{a}_{(i^*)}$, $b_L = b_{(i^*)}$,
           and $\beta_L = \beta_{(i^*)}$.
       endwhile
```

## 3 Performance Evaluation

In this section, the performance of the proposed enhanced incremental learning algorithm is compared with the original I-ELM on benchmark regression problems from UCI database [25]. The specification of these benchmark problems are shown in Table 1. In our experiments, all the inputs (attributes) have been normalized into the range $[-1, 1]$ while the outputs (targets) have been normalized into $[0, 1]$. Neural networks are tested in I-ELM and its enhancements EI-ELM. Popular additive and RBF hidden nodes in the real domain space (a specific case of the complex domain) are used. It should be noted that similar results can be obtained in the complex domain as well. For the sigmoid type of additive hidden nodes $G(\mathbf{x}, \mathbf{a}, b) = \frac{1}{1+e^{-(\mathbf{a}\cdot\mathbf{x}+b)}}$ the hidden parameters $\mathbf{a}$ and $b$ are randomly chosen from the range $[-1, 1]$. For the RBF hidden nodes $G(\mathbf{x}, \mathbf{a}, b) = \exp(-b\|\mathbf{x} - \mathbf{a}\|^2)$, the hidden node parameters $\mathbf{a}$ are randomly chosen from the range $[-1, 1]$ whereas the hidden node parameter $b$ is randomly generated from the range $(0, 0.5)$. The target error is set as $\epsilon = 0.01$. All the simulations are running in MATLAB 6.5 environment and the same PC with Pentium 4 2.99 GHZ CPU and 1G RAM. For each problem, the average results over 20 trials are obtained for EI-ELM and I-ELM.

| Name | No. of Observations | | Attributes |
|---|---|---|---|
| | Training data | Testing data | |
| Abalone | 2000 | 2177 | 8 |
| Ailerons | 7154 | 6596 | 39 |
| Airplane | 450 | 500 | 9 |
| Auto Price | 80 | 79 | 15 |
| Bank | 4500 | 3692 | 8 |
| Boston | 250 | 256 | 13 |
| California | 8000 | 12640 | 8 |
| Census (House8L) | 10000 | 12784 | 8 |
| Computer Activity | 4000 | 4192 | 12 |
| Delta Ailerons | 3000 | 4129 | 5 |
| Delta Elevators | 4000 | 5517 | 6 |
| Kinematics | 4000 | 4192 | 8 |
| Machine CPU | 100 | 109 | 6 |
| Puma | 4500 | 3692 | 8 |
| Pyrim | 40 | 34 | 26 |
| Servo | 80 | 87 | 4 |

Table 1
Specification of 16 Benchmarking Regression Datasets

### *3.1 Comparison between EI-ELM and I-ELM with the same number of hidden nodes*

We first compare the generalization performance of EI-ELM and I-ELM with the same number of hidden nodes. Table 2 and Table 3 show the performance of EI-ELM and I-ELM with 200 additive and RBF nodes respectively. Seen from Table 2 and Table 3, the testing root mean square error (RMSE) of EI-ELM are generally much smaller than that of the original I-ELM when both EI-ELM and I-ELM use the same number of hidden nodes. It shows that EI-ELM may achieve faster convergence rate than I-ELM under the same nodes. This can also be verified from Fig. 1 which shows testing error curves of EI-ELM and I-ELM for both Airplane and Census cases. Furthermore, as observed from Table 2 and Table 3, EI-ELM obtain much smaller standard deviation of testing RMSE than I-ELM. This means that the performance of EI-ELM is stabler than I-ELM.

| Name | EI-ELM (Sigmoid, $k = 10$) | | | I-ELM (Sigmoid, $k = 1$) | | |
|---|---|---|---|---|---|---|
| | Mean | Dev | Time(s) | Mean | Dev | Time(s) |
| Abalone | **0.0818** | 0.0020 | 2.5801 | 0.0920 | 0.0046 | 0.2214 |
| Ailerons | **0.0558** | 0.0024 | 9.5017 | 0.1023 | 0.0353 | 0.7547 |
| Airplane | **0.0804** | 0.0039 | 0.5669 | 0.1016 | 0.0093 | 0.0499 |
| Auto Price | **0.0896** | 0.0022 | 0.3141 | 0.0977 | 0.0069 | 0.0329 |
| Bank | **0.0631** | 0.0031 | 3.9838 | 0.1173 | 0.0068 | 0.3237 |
| Boston | **0.1055** | 0.0098 | 0.4332 | 0.1167 | 0.0112 | 0.0515 |
| California | **0.1494** | 0.0015 | 9.3336 | 0.1683 | 0.0049 | 0.5448 |
| Census (8L) | **0.0829** | 0.0012 | 11.476 | 0.0923 | 0.0023 | 0.8667 |
| Computer Activity | **0.0941** | 0.0028 | 3.7511 | 0.1201 | 0.0125 | 0.2794 |
| Delta Ailerons | **0.0445** | 0.0062 | 2.7735 | 0.0525 | 0.0078 | 0.2620 |
| Delta Elevators | **0.0582** | 0.0032 | 3.7971 | 0.0740 | 0.0126 | 0.2708 |
| Kinematics | <u>0.1393</u> | 0.0028 | 3.4373 | <u>0.1418</u> | 0.0033 | 0.2810 |
| Machine CPU | <u>0.0466</u> | 0.0060 | 0.3112 | <u>0.0504</u> | 0.0079 | 0.0234 |
| Puma | <u>0.1840</u> | 0.0017 | 3.9531 | <u>0.1861</u> | 0.0041 | 0.3236 |
| Pyrim | **0.1414** | 0.0341 | 0.3062 | 0.1867 | 0.0628 | 0.0374 |
| Servo | **0.1518** | 0.0116 | 0.3235 | 0.1662 | 0.0124 | 0.0218 |

Table 2
Performance comparison between EI-ELM and I-ELM (both with 200 Sigmoid hidden nodes)

The training time curves for several cases are shown in Fig 2 (for Sigmoid additive node case). Seen from Fig 2, the spent learning time is still linearly increasing with the learning steps, which is the same as I-ELM's conclusion. Since the training time of I-ELM linearly increases with the number of hidden nodes, in theory the training time of EI-ELM should be around $k$ times that of I-ELM, where $k$ is optimal trial times. In our simulations, we set $k = 10$. The training time of EI-ELM should be almost 10 times of I-ELM. However, EI-ELM needs to seek for the most appropriate parameters from $k$ trials, furthermore, for all the applications, we need to store some temp variables, which normally use large memory. When memory reach computer's choke-point, it will disturb CPU's computation remarkably. Especially for applications with large scale data, the influence becomes more obvious. So the training time relationship between EI-ELM and I-ELM should be that the training time

| Name | EI-ELM (RBF, $k=10$) | | | I-ELM (RBF, $k=1$) | | |
|---|---|---|---|---|---|---|
| | Mean | Dev | Time(s) | Mean | Dev | Time(s) |
| Abalone | **0.0829** | 0.0027 | 5.6006 | 0.0938 | 0.0053 | 0.5030 |
| Ailerons | **0.0774** | 0.0129 | 36.016 | 0.1430 | 0.0298 | 3.2769 |
| Airplane | **0.0633** | 0.0057 | 0.9578 | 0.0992 | 0.0166 | 0.0751 |
| Auto Price | **0.1139** | 0.0189 | 0.4031 | 0.1261 | 0.0255 | 0.0468 |
| Bank | **0.0730** | 0.0022 | 9.8079 | 0.1157 | 0.0097 | 0.7782 |
| Boston | **0.1077** | 0.0084 | 0.7972 | 0.1320 | 0.0126 | 0.0657 |
| California | **0.1503** | 0.0022 | 17.133 | 0.1731 | 0.0081 | 1.3656 |
| Census (8L) | **0.0810** | 0.0016 | 19.922 | 0.0922 | 0.0029 | 1.7928 |
| Computer Activity | **0.1153** | 0.0021 | 10.092 | 0.1552 | 0.0282 | 0.8220 |
| Delta Ailerons | **0.0448** | 0.0065 | 4.6169 | 0.0632 | 0.0116 | 0.4327 |
| Delta Elevators | **0.0575** | 0.0047 | 7.3541 | 0.0790 | 0.0123 | 0.6321 |
| Kinematics | **0.1213** | 0.0017 | 8.3114 | 0.1555 | 0.0122 | 0.6953 |
| Machine CPU | **0.0554** | 0.0148 | 0.4114 | 0.0674 | 0.0177 | 0.0447 |
| Puma | **0.1752** | 0.0022 | 9.7983 | 0.1913 | 0.0180 | 0.7872 |
| Pyrim | **0.1209** | 0.0431 | 0.4423 | 0.2241 | 0.1752 | 0.0434 |
| Servo | **0.1379** | 0.0151 | 0.4031 | 0.1524 | 0.0200 | 0.0391 |

Table 3
Performance comparison between EI-ELM and I-ELM (both with 200 RBF hidden nodes)

of EI-ELM should be $k + \eta$ times of I-ELM, where $\eta$ is impact factor determined by computing environment, etc. The training time comparisons shown in Table 2 and Table 3 have verified our conclusion.

### *3.2 Comparison between EI-ELM and I-ELM with the different number of hidden nodes*

During our simulations we also studied the case where EI-ELM and I-ELM are given different number of hidden nodes. The comparison between EI-ELM with 50 hidden nodes and I-ELM with 500 hidden nodes are shown in Table 4 and Table 5. Here we set $k = 10$ and $k = 20$ for EI-ELM respectively. As observed from Table 4 and Table 5, the generalization performance obtained by EI-ELM with 50 hidden nodes are usually slightly better than those obtained by I-ELM with 500 hidden nodes. It further demonstrates that EI-ELM can

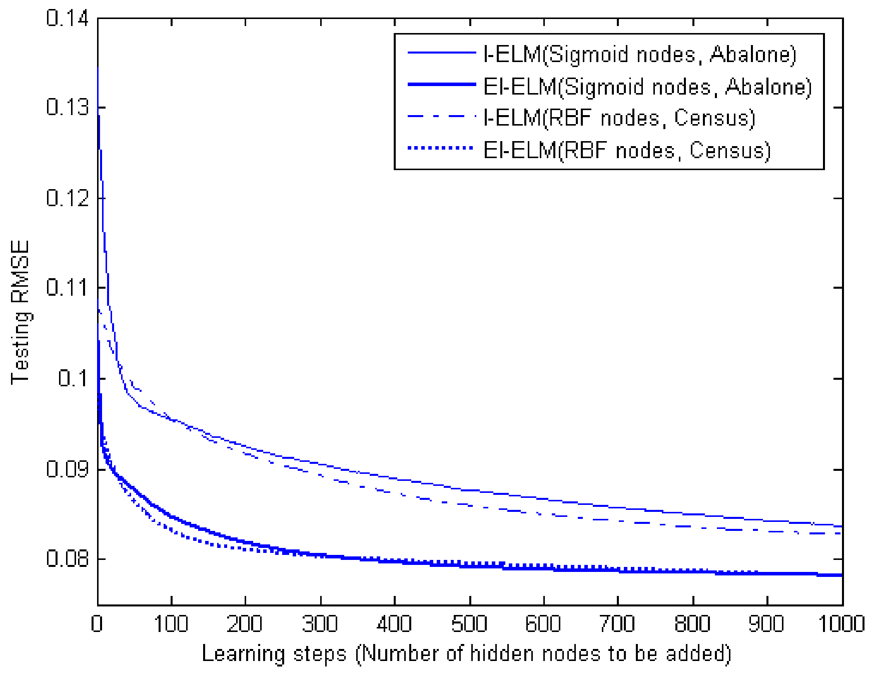

Figure 1. The testing error updating curves of EI-ELM and I-ELM

converge faster than I-ELM. It can also be seen that the training time spent by I-ELM with 500 hidden nodes and EI-ELM with 50 hidden nodes are in the same order when $k = 10$, which is consistent with our previous analysis on the training time of EI-ELM. However compact network architecture implies faster prediction time. It can be seen that EI-ELM with only 50 hidden nodes can achieve the similar results as I-ELM with 500 hidden nodes do. Fig 3 shows the testing error curves of I-ELM and EI-ELM with sigmoid activation functions for the Abalone case. It can be seen that in order to obtain the same testing RMSE 0.09, EI-ELM only needs 21 nodes and spends 0.2722 seconds on training while I-ELM needs 331 nodes and spends 0.5231 seconds on training. Similar curves have been found for other regression cases.

### *3.3 Effects of number of selecting trial factor k*

Effects of number of selecting trials $k$ on the performance of EI-ELM have also been investigated in our study. Fig. 4 and Fig. 5 show that the generalization performance and stability of EI-ELM with 100 sigmoid hidden nodes will become better when the number of selecting trials $k$ increases for the Airplane case. Fig 6 shows the testing RMSE curves of EI-ELM with increasing number of hidden nodes and different number of selecting trials $k$. It can be seen that the effect of number of selecting trials $k$ on the generalization performance and its stability tend to become stable after $k$ increases up to certain number

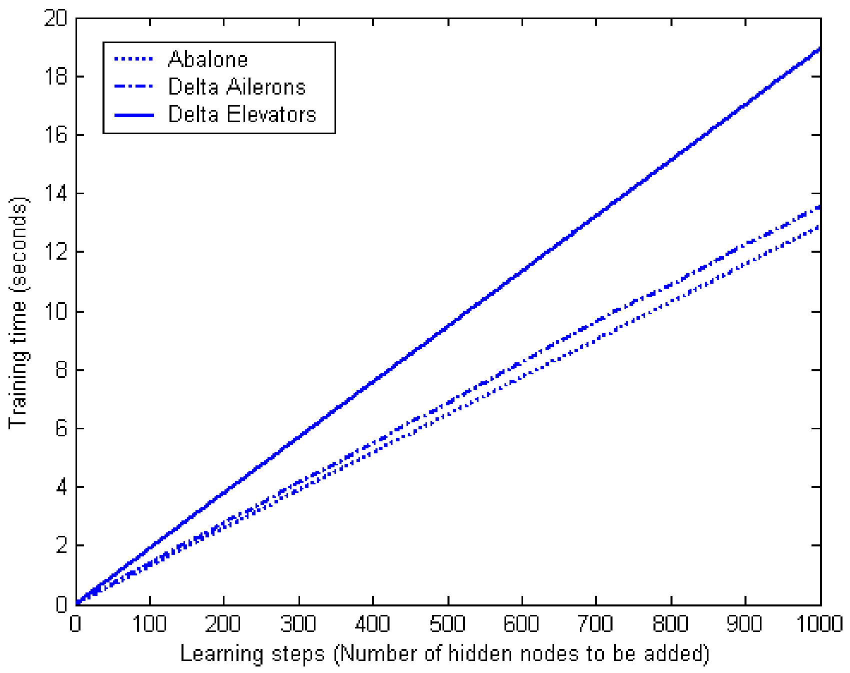

Figure 2. Training time spent by EI-ELM is linearly increased with the number of hidden nodes to be added

(around 20 in Airplane case). The above observations are true for the rest cases as well.

## 4 Conclusion

In this paper, inspired by the latest theory [1], we proposed an enhanced random search based incremental algorithm EI-ELM for I-ELM. Similar to I-ELM, EI-ELM also randomly generates hidden nodes and then analytically determines the output weights. EI-ELM works for different type of hidden node output functions instead of neural networks only. It should be noted that I-ELM is a specific case of EI-ELM when $k = 1$. The difference between I-ELM and EI-ELM is that at each learning step among several randomly generated hidden nodes EI-ELM picks the optimal hidden node which leads to the smallest residual error. Compared with the original I-ELM, EI-ELM can achieve faster convergence rate and much more compact network architectures, which have been further verified by the simulation results on some benchmark real-world regression problems. Reasonably speaking, EI-ELM could also be applied to the convex optimization based incremental extreme learning machine (CI-ELM)[20] straightforward, the performance of EI-ELM with convex optimization is worth investigating in the future.

| Problems | EI-ELM (50 Sigmoid hidden nodes) | | | | | | I-ELM (500 Sigmoid hidden nodes, $k = 1$) | | |
|---|---|---|---|---|---|---|---|---|---|
| | $k = 10$ | | | $k = 20$ | | | | | |
| | Mean | Dev | Time(s) | Mean | Dev | Time(s) | Mean | Dev | Time(s) |
| Abalone | 0.0878 | 0.0033 | 0.6506 | 0.0876 | 0.0015 | 1.5785 | 0.0876 | 0.0033 | 0.7695 |
| Ailerons | 0.0640 | 0.0066 | 2.3766 | **0.0571** | 0.0022 | 6.2519 | 0.0824 | 0.0232 | 1.8810 |
| Airplane | 0.0922 | 0.0061 | 0.1389 | 0.0862 | 0.0040 | 0.2921 | 0.0898 | 0.0067 | 0.1466 |
| Auto Price | 0.0924 | 0.0112 | 0.0814 | **0.0897** | 0.0104 | 0.1658 | 0.0948 | 0.0158 | 0.0561 |
| Bank | 0.1066 | 0.0058 | 0.9965 | 0.0896 | 0.0036 | 3.1058 | **0.0757** | 0.0032 | 0.7914 |
| Boston | 0.1133 | 0.0101 | 0.1065 | 0.1102 | 0.0061 | 0.2232 | 0.1084 | 0.0096 | 0.1033 |
| California | 0.1591 | 0.0034 | 2.2423 | 0.1548 | 0.0033 | 4.9486 | 0.1543 | 0.0019 | 1.5665 |
| Census (8L) | 0.0899 | 0.0017 | 2.8655 | 0.0865 | 0.0011 | 6.1100 | 0.0871 | 0.0018 | 2.1199 |
| Computer Activity | 0.1075 | 0.0057 | 0.9342 | **0.0991** | 0.0036 | 2.3311 | 0.1057 | 0.0078 | 0.7185 |
| Delta Ailerons | 0.0474 | 0.0062 | 0.7006 | 0.0467 | 0.0042 | 1.4570 | 0.0468 | 0.0052 | 0.6340 |
| Delta Elevators | 0.0615 | 0.0049 | 0.9502 | **0.0586** | 0.0038 | 2.5385 | 0.0640 | 0.0055 | 0.6516 |
| Kinematics | 0.1420 | 0.0029 | 0.8655 | 0.1416 | 0.0019 | 2.9017 | 0.1406 | 0.0014 | 0.7117 |
| Machine CPU | 0.0498 | 0.0155 | 0.0750 | 0.0467 | 0.0148 | 0.1577 | 0.0474 | 0.0040 | 0.0645 |
| Puma | 0.1846 | 0.0018 | 0.9856 | 0.1827 | 0.0017 | 2.7264 | 0.1856 | 0.0039 | 0.7983 |
| Pyrim | 0.1514 | 0.0419 | 0.0782 | **0.1300** | 0.0405 | 0.1533 | 0.1712 | 0.0626 | 0.0810 |
| Servo | 0.1634 | 0.0129 | 0.0795 | 0.1558 | 0.0121 | 0.1611 | 0.1589 | 0.0124 | 0.0642 |

Table 4
Performance comparison between EI-ELM with 50 Sigmoid hidden nodes and I-ELM with 500 Sigmoid hidden nodes

## Acknowledgment

We would like to thank Halbert White, University of California, San Diego, US for the helpful discussions and kind clarification on his significant seminal works.

## References

[1] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[2] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine," in *the IASTED International Conference on Computational Intelligence (CI 2005)*, (Calgary, Canada), July 4-6, 2005.

[3] T. Kim and T. Adali, "Approximation by fully complex multilayer perseptrons," *Neural Computation*, vol. 15, pp. 1641–1666, 2003.

[4] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, pp. 306–314, 2005.

| Problems | EI-ELM (50 RBF hidden nodes) | | | | | | I-ELM (500 RBF hidden nodes, $k = 1$) | | |
|---|---|---|---|---|---|---|---|---|---|
| | $k = 10$ | | | $k = 20$ | | | | | |
| | Mean | Dev | Time(s) | Mean | Dev | Time(s) | Mean | Dev | Time(s) |
| Abalone | 0.0907 | 0.0034 | 1.4036 | 0.0871 | 0.0023 | 3.0006 | 0.0872 | 0.0022 | 1.2121 |
| Ailerons | 0.0973 | 0.0229 | 9.0306 | **0.0775** | 0.0033 | 19.071 | 0.1129 | 0.0295 | 8.1818 |
| Airplane | 0.0943 | 0.0168 | 0.2347 | 0.0813 | 0.0102 | 0.5487 | 0.0772 | 0.0082 | 0.1940 |
| Auto Price | 0.1187 | 0.0159 | 0.0998 | **0.1104** | 0.0148 | 0.2110 | 0.1231 | 0.0133 | 0.1189 |
| Bank | 0.0989 | 0.0031 | 2.4460 | 0.0888 | 0.0023 | 5.4199 | 0.0843 | 0.0058 | 1.9382 |
| Boston | 0.1197 | 0.0107 | 0.1845 | **0.1171** | 0.0078 | 0.3621 | 0.1214 | 0.0103 | 0.1872 |
| California | 0.1624 | 0.0049 | 4.2339 | 0.1579 | 0.0027 | 8.8326 | 0.1582 | 0.0027 | 3.8482 |
| Census (8L) | 0.0864 | 0.0026 | 4.9858 | 0.0846 | 0.0020 | 11.796 | 0.0860 | 0.0018 | 4.8536 |
| Computer Activity | 0.1295 | 0.0068 | 2.4905 | **0.1201** | 0.0024 | 5.5878 | 0.1358 | 0.0177 | 2.1267 |
| Delta Ailerons | 0.0469 | 0.0067 | 1.1800 | **0.0466** | 0.0039 | 2.4763 | 0.0544 | 0.0076 | 1.0361 |
| Delta Elevators | 0.0603 | 0.0049 | 1.8515 | **0.0602** | 0.0039 | 4.2506 | 0.0685 | 0.0099 | 1.5399 |
| Kinematics | 0.1346 | 0.0025 | 2.0913 | **0.1306** | 0.0019 | 4.6727 | 0.1425 | 0.0095 | 1.7042 |
| Machine CPU | 0.0622 | 0.0281 | 0.1067 | **0.0511** | 0.0114 | 0.2031 | 0.0614 | 0.0274 | 0.0875 |
| Puma | 0.1789 | 0.0020 | 2.4465 | **0.1770** | 0.0012 | 5.2821 | 0.1850 | 0.0119 | 1.9709 |
| Pyrim | 0.1214 | 0.0345 | 0.1016 | **0.0989** | 0.0286 | 0.2079 | 0.2179 | 0.1545 | 0.1071 |
| Servo | 0.1487 | 0.0133 | 0.0985 | 0.1434 | 0.0120 | 0.1958 | 0.1410 | 0.0151 | 0.0982 |

Table 5
Performance comparison between EI-ELM with 50 RBF hidden nodes and I-ELM with 500 RBF hidden nodes

[5] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 999–1013, 1993.

[6] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," *Applied Optics*, vol. 26, no. 23, pp. 4972–4978, 1987.

[7] Y. Shin and J. Ghosh, "Approximation of multivariate functions using ridge polynomial networks," in *Proceedings of International Joint Conference on Neural Networks (IJCNN'2002)*, (Baltimore, MD, USA), pp. 380–385, June 2002.

[8] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, pp. 909–996, 1988.

[9] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[10] F. Han and D.-S. Huang, "Improved extreme learning machine for function approximation by encoding a priori information," *Neurocomputing*, vol. 69, pp. 2369–2373, 2006.

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

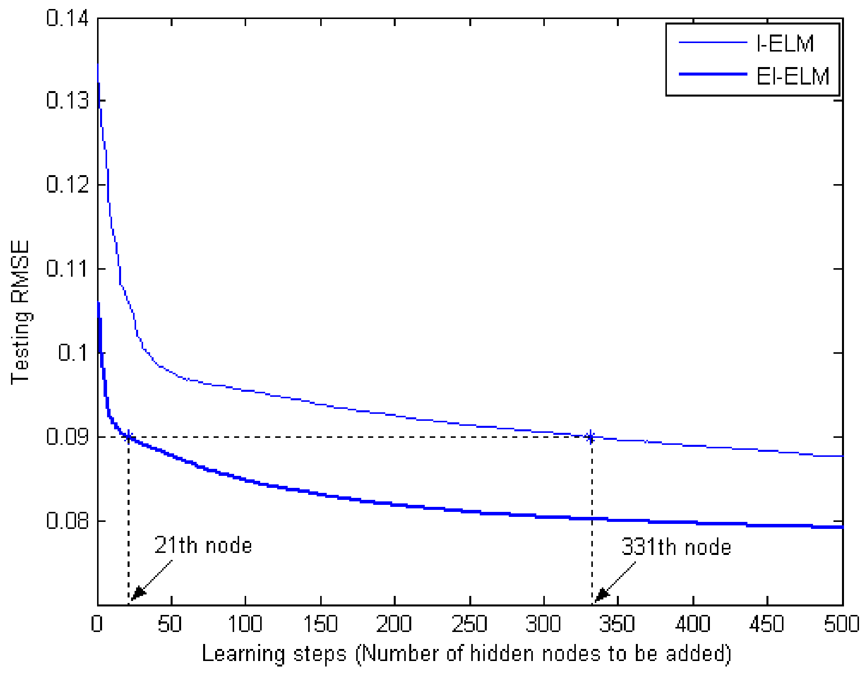

Figure 3. Testing RMSE performance comparison between EI-ELM and I-ELM (with Sigmoid hidden nodes) for Abalone case

[12] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246–257, 1991.

[13] H. White, "An additional hidden unit test for neglected nonlinearity in multilayer feedforward networks," in *Proceedings of the International Conference on Neural Networks*, pp. 451–455, 1989.

[14] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004)*, vol. 2, (Kunming, China), pp. 1029–1036, 6-9 Dec, 2004.

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

[16] G.-B. Huang, Q.-Y. Zhu, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?," *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187–191, 2006.

[17] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Real-time learning capability of neural networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 863–878, 2006.

[18] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate on-line sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

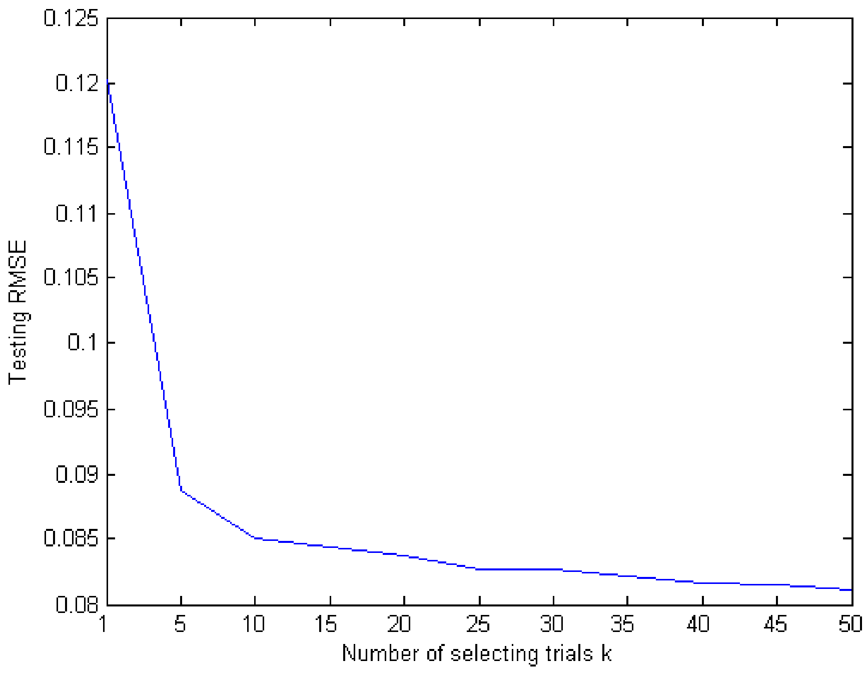

Figure 4. Effect of number of selecting trials $k$ on the generalization performance of EI-ELM in Airplane case

[19] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *(in press) Neurocomputing*, 2007.

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

[21] T.-Y. Kwok and D.-Y. Yeung, "Objective functions for training new hidden units in constructive neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 1131 –1148, 1997.

[22] T.-H. Lee, H. White, and C. W. J. Granger, "Testing for neglected nonlinearity in time series modes: A comparison of neural network methods and standard tests," *Journal of Econometrics*, vol. 56, pp. 269–290, 1993.

[23] M. B. Stinchcombe and H. White, "Consistent specification testing with nuisance parameters present only under the alternative," *Econometric Theory*, vol. 14, pp. 295–324, 1998.

[24] H. White, "Approxiate nonlinear forecasting methods," in *Handbook of Economics Forecasting* (G. Elliott, C. W. J. Granger, and A. Timmermann, eds.), pp. 460–512, New York: Elsevier, 2006.

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

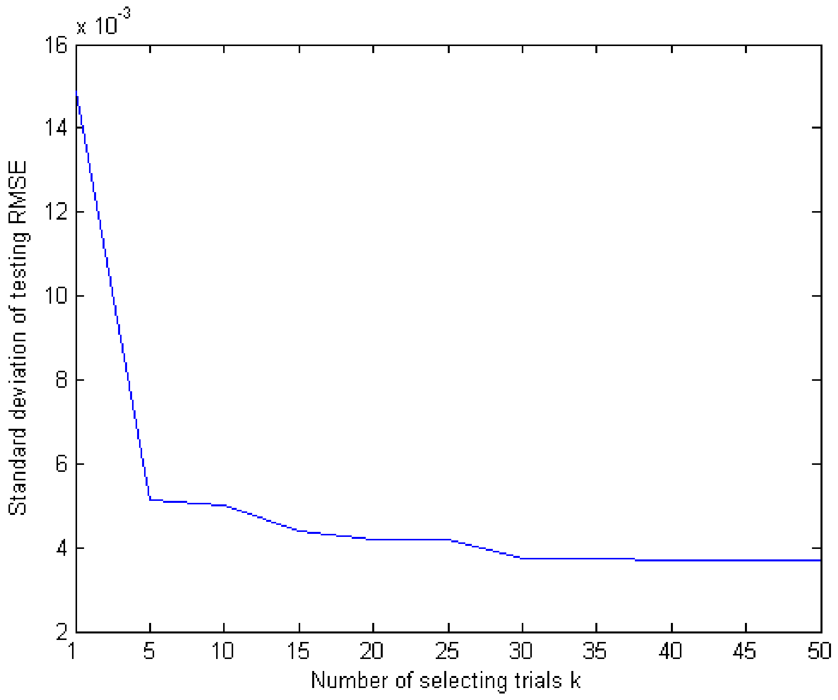

Figure 5. Effect of number of selecting trials $k$ on the stability of the generalization performance of EI-ELM in Airplane case

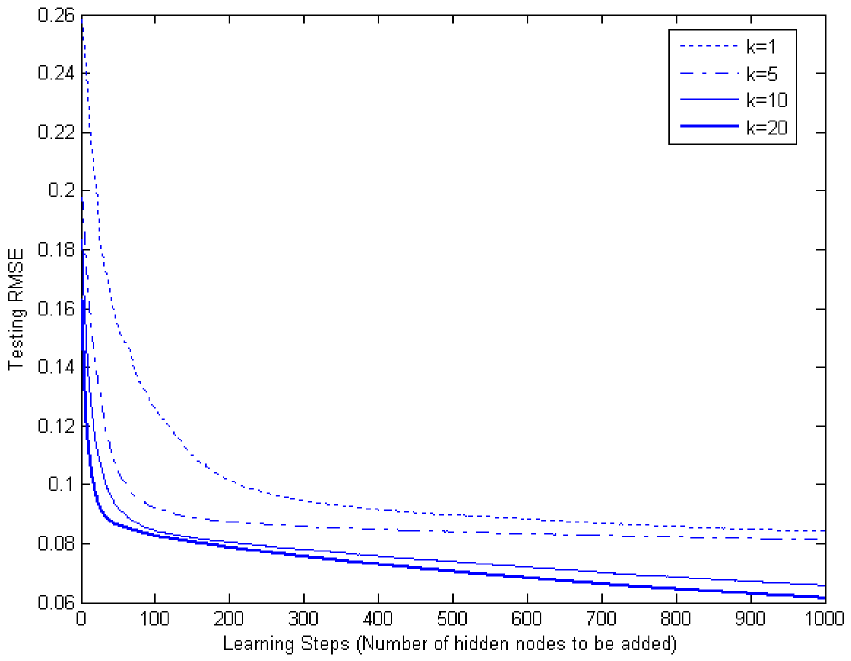

Figure 6. Testing RMSE updating progress with new hidden nodes added and different number of selecting trials $k$ in Airplane case