Brief Papers

Bidirectional Extreme Learning Machine for Regression Problem and Its Learning Effectiveness

Yimin Yang, Yaonan Wang, and Xiaofang Yuan

***Abstract*— It is clear that the learning effectiveness and learning speed of neural networks are in general far slower than required, which has been a major bottleneck for many applications. Recently, a simple and efficient learning method, referred to as extreme learning machine (ELM), was proposed by Huang *et al.*, which has shown that, compared to some conventional methods, the training time of neural networks can be reduced by a thousand times. However, one of the open problems in ELM research is whether the number of hidden nodes can be further reduced without affecting learning effectiveness. This brief proposes a new learning algorithm, called bidirectional extreme learning machine (B-ELM), in which some hidden nodes are not randomly selected. In theory, this algorithm tends to reduce network output error to 0 at an extremely early learning stage. Furthermore, we find a relationship between the network output error and the network output weights in the proposed B-ELM. Simulation results demonstrate that the proposed method can be tens to hundreds of times faster than other incremental ELM algorithms.**

***Index Terms*— Feedforward neural network, learning effectiveness, number of hidden nodes, universal approximation.**

I. Introduction

As a specific type of neural network, the single-hidden-layer feedforward network (SLFN) plays an important role in practical applications. Unlike conventional neural network theories, Huang *et al.* [1]–[6] have recently proved that SLFNs with additive or radial basis function hidden nodes and with randomly generated hidden node parameters can work as universal approximators by only calculating the output weights linking the hidden layer to the output nodes. In [3] and [7], Huang *et al.* demonstrated that iterative techniques are not required for adjusting the parameters of SLFNs at all. Based on the universal approximation capability of SLFNs with random hidden nodes, Huang *et al.* in [2] and [8], respectively, proposed simple and efficient learning steps both with increased network architecture incremental extreme learning machine (I-ELM), and with fixed-network architecture (ELM). These two methods generate parameters of a hidden node randomly, so training the SLFNs simply amounts to getting the output weight. This makes the selection of the weights of the hidden neurons very fast in the case of SLFNs. Hence, the overall computational time for model structure selection and actual training of the model is often reduced by several hundred times than in some conventional methods such as BP. Based on the ELM with incremental network architecture [8], methods with the mechanism of growth hidden nodes were proposed, such as enhanced incremental ELM (EI-ELM) [9], optimal pruned ELM (OP-ELM) [10], convex incremental ELM (CI-ELM) [11], and error-minimized LM (EM-ELM) [12], to obtain better performance or to reduce the network structure (the number of hidden nodes). Also, based on ELM with a fixed network architecture [2], methods were developed with a fixed network architecture, such as online sequential learning model ELM (OS-ELM) [7] and online sequential fuzzy learning model ELM (OS-FNN-ELM) [13].

Manuscript received October 4, 2011; accepted May 24, 2012. Date of publication June 20, 2012; date of current version August 1, 2012. This work was supported in part by the National Natural Science Foundation of China under Grant 61175075, Grant 60835004, and Grant 61104088, and the National Technology Support under Project 2008BAF36B01.

The authors are with the College of Electrical and Information Engineering, Hunan University, Changsha 410082, China (e-mail: yangyi_min@126.com; yaonan@hnu.edu.cn; yuanxiaof@21cn.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2202289

Although the network training utilizing ELM or I-ELM is faster than other algorithms, there are still two major unresolved problems [5].

1) For ELMs with fixed network architecture, the only factor that needs to be set by users is the size of the SLFNs (the number of hidden nodes). However, how to choose the optimum number of hidden nodes is still unknown, it is usually done by trial and error. For example, in Fig. 1, we can find that the best testing root-mean-square error (RMSE) is about 0.05 with 16 hidden nodes. However, when the number of hidden nodes is increased to 25, the average testing RMSE increased to 0.07. This is because in [2], Huang *et al.* did not prove that the residual error of SLFNs is a strictly monotonic sequence when the number of hidden nodes increases.

2) For incremental ELM methods such as I-ELM, EI-ELM, EM-ELM, and OP-ELM, Huang *et al.* proved in [8] that with increasing hidden nodes, the residual error of SLFNs decreased and bounded below by zero. Unlike ELM with a fixed network architecture, this result makes it easy for the user to determine the network structure by adding hidden nodes one by one until achieving the expected training accuracy. However, experimental results show that the learning time of incremental ELM methods increase many times compared to that of ELM. It is because I-ELM calculates n output weights one by one when n hidden nodes are used. But ELM only calculates these n output weights once when n hidden nodes are used.

Based on I-ELM, this brief proposes a new incremental learning algorithm named *bidirectional extreme learning machine* (B-ELM). The following are the contributions of this brief.

1) The learning of the proposed B-ELM can be tens to hundreds of times faster than other incremental ELM

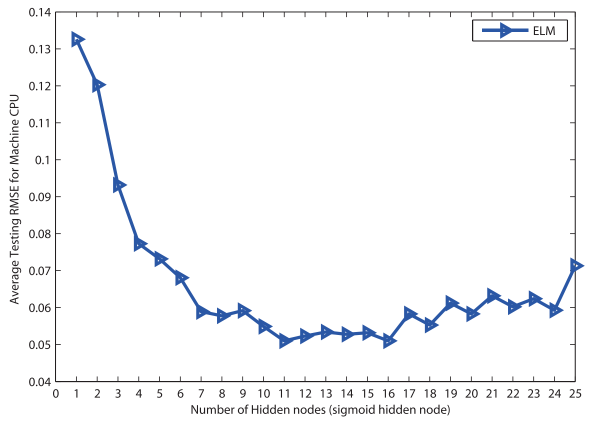

Fig. 1. Averaged testing RMSE of ELM over 50 experiments in the Machine CPU dataset.

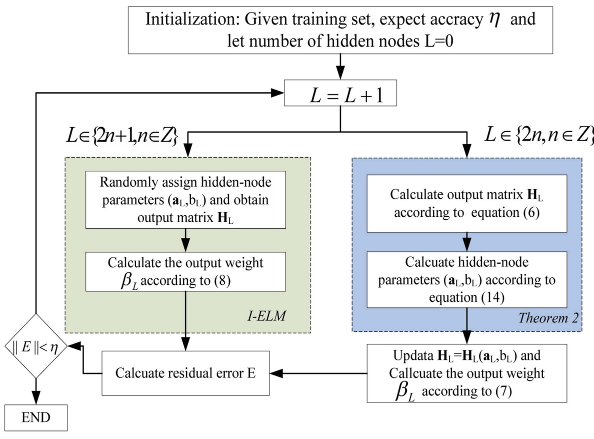
Initialization: Given training set, expect accracy η and let number of hidden nodes L=0
$L=L+1$
$L\in\{2n+1, n\in Z\}$
$L\in\{2n, n\in Z\}$
Randomly assign hidden-node parameters ($\mathbf{a}_L$,b_L) and obtain output matrix $\mathbf{H}_L$
Calculate the output weight β_L according to (8)
I-ELM
Calculate output matrix $\mathbf{H}_L$ according to equation (6)
Calcuate hidden-node parameters ($\mathbf{a}_L$,b_L) according to equation (14)
Theorem 2
$\|E\|<\eta$
Calcuate residual error E
Updata $\mathbf{H}_L$=$\mathbf{H}_L$($\mathbf{a}_L$,b_L) and Callcuate the output weight β_L according to (7)
END

Fig. 2. Structure of proposed B-ELM.

methods. Furthermore, simulation results show that B-ELM with only two hidden nodes can achieve similar generalization performance as the I-ELM with hundreds of hidden nodes. This means B-ELM can reduce the number of hidden nodes hundreds of times compared to I-ELM.

2) For B-ELM, we find a relationship between the residual error function e and network output weights β and we named this relationship the *error-output weights ellipse*: $\beta_{2n}^2/\beta_{2n-1}^2 + \|e_{2n}\|^2/\|e_{2n-1}\|^2 = 1$, where e_i denotes the residual error function for the current network with i hidden nodes and β_i is the weight connecting the ith hidden node to the output node. This expression implies that the learning effectiveness only depends on $\beta_{2n}^2/\beta_{2n-1}^2$. If $|\beta_{2n}|/|\beta_{2n-1}| \to 1$, $\|e_{2n}\| << \|e_{2n-1}\|$.

3) Inspired by the anonymous reviewers' comments, we prove that B-ELM with two hidden nodes can reduce the residual error e_2 to 0 under the condition of $f = T$, where f and T are defined in Section II. This means B-ELM calculates the hidden weights only once and then SLFNs can approximate to any continuous target function. In other words, in B-ELM, "one iteration" on the hidden layer can give significant improvements on accuracy instead of maintaining a completely random hidden layer.

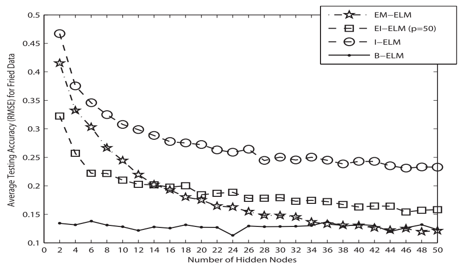

Fig. 3. Average testing RMSE of different algorithms in Fried case (sine hidden nodes).

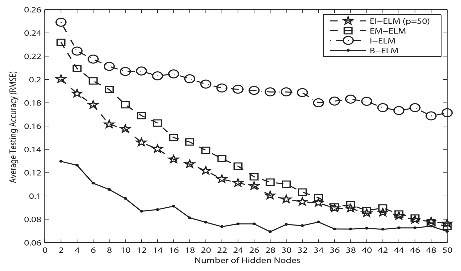

Fig. 4. Average testing RMSE of different algorithms in Bank case (sine hidden nodes).

II. Preliminaries and Notation

A. Notations and Definitions

The sets of real, integer, positive real, and positive integer numbers are denoted by $\mathbf{R}, \mathbf{Z}, \mathbf{R}^+$, and $\mathbf{Z}^+$, respectively. Similar to [8], let $\mathcal{L}^2(X)$ be a space of functions f on a compact subset X in the n-dimensional Euclidean space $\mathbf{R}^n$ such that $|f|^2$ are integrable, i.e., $\int_X |f(x)|^2 dx < \infty$. Let $\mathcal{L}^2(\mathbf{R}^n)$ be denoted by $\mathcal{L}^2$. For $u, v \in \mathcal{L}^2(X)$, the inner product $< u, v >$ is defined by

$$< u, v > = \int_X u(\mathbf{x})\overline{v(\mathbf{x})} d\mathbf{x}. \tag{1}$$

The norm in $\mathcal{L}^2(X)$ space will be denoted as $\|\cdot\|$. L denotes the number of hidden nodes. N denotes the number of arbitrary distinct samples. $\mathbf{H}$ is called the hidden layer output matrix of the SLFNs, the ith column of $\mathbf{H}$ ($\mathbf{H}_i$) is the ith hidden node output with respect to inputs. The hidden layer output matrix $\mathbf{H}_i$ is said to be a randomly generated function sequence $\mathbf{H}_i^r$ if the corresponding hidden-node parameters $(\mathbf{a}_i, b_i)$ are randomly generated. e_n denotes the residual error function for the current network f_n with n hidden nodes. For N training samples $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$, T denotes $T = [t_1, \ldots, t_n]$ and $e_i(p)$ is the residual error for the input of pth training sample with i hidden nodes and $E_i = [e_i(1), \ldots, e_i(N)]$ is the residual vector.

B. I-ELM

For N arbitrary distinct samples $(\mathbf{x}_i, t_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $t_i \in \mathbf{R}$, standard SLFNs with L hidden nodes and activation function $h(x)$ are mathematically modeled as

$$\sum_{i=1}^{n} \beta_i h(\mathbf{a}_i \cdot \mathbf{x}_j + b_i) = t_j, \mathbf{a}_i \in \mathbf{R}^n, \quad b_i, \beta_i \in \mathbf{R}, j = 1, \ldots, N \tag{2}$$

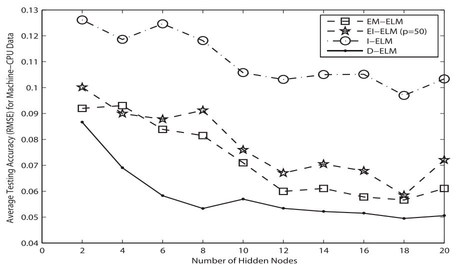

Fig. 5. Average testing RMSE of different algorithms in Machine CPU case (sine hidden nodes).

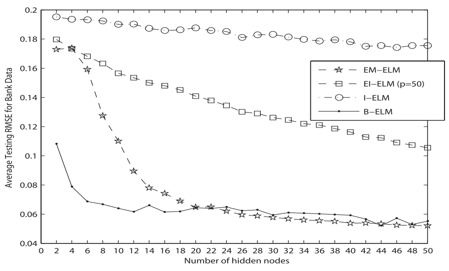

Fig. 7. Average testing RMSE of different algorithms in Bank case (sigmoid hidden nodes).

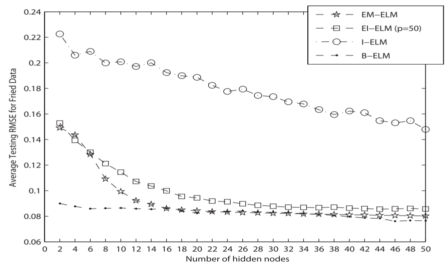

Fig. 6. Average testing RMSE of different algorithms in Fried case (sigmoid hidden nodes).

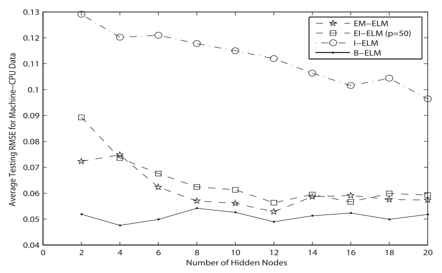

Fig. 8. Average testing RMSE of different algorithms in Machine CPU case (sigmoid hidden nodes).

where $\mathbf{a}_i$ is the weight vector connecting the input layer to the ith hidden node, b_i is the bias of the ith hidden node, β_i is the weight connecting the ith hidden node to the output node, and h is the hidden node activation function. $\mathbf{a}_i \cdot \mathbf{x}$ denotes the inner product of vector $\mathbf{a}_i$ and $\mathbf{x}$ in $\mathbf{R}^n$.

Equation (2) can be written compactly as

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

where $T = [t_1, \ldots, t_N]$. The closeness between network function f_n and the target function f is measured by the F^2 distance

$$\|f_n - f\| = \left[\int_X (f_n(\mathbf{x}) - f(\mathbf{x}))\overline{(f_n(\mathbf{x}) - f(\mathbf{x}))}d\mathbf{x}\right]^{\frac{1}{2}}. \tag{4}$$

The I-ELM 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. Thus, training SLFNs simply amounts to getting the output weights β. Denote the residual error of f_n as $e_n = f - f_n$. Huang *et al.* have proved the following lemma.

Lemma 1 ([8]): Given an SLFN with nonconstant piecewise continuous hidden nodes $\mathbf{H}(\mathbf{x}, \mathbf{a}, b)$, then for any continuous target function f and any function sequence $\mathbf{H}_n^r(\mathbf{x}) = \mathbf{H}(\mathbf{x}, \mathbf{a}_n, b_n)$ randomly generated based on any continuous sampling distribution, $\lim_{n\to\infty} \|f - (f_{n-1} + \mathbf{H}_n^r \beta_n)\| = 0$ holds with probability 1 if

$$\beta_n = \frac{\langle e_{n-1}, \mathbf{H}_n^r\rangle}{\|\mathbf{H}_n^r\|^2} \tag{5}$$

where $e_n = f - f_n$ denotes the residual error function for the current network f_n with n hidden nodes.

III. Proposed Bidirectional ELM Method

In this section, we first indicate the structure of the proposed bi-direction ELM method in Section III-A. Then in Section III-B, we prove that SLFNs with hidden nodes generated by B-ELM and with sine or sigmoid activation function can universally approximate any continuous target functions in any compact subset. Then we discuss the learning effectiveness of B-ELM, and, finally, the pseudo-code of proposed B-ELM is given in Section III-C.

A. Structure of the Proposed Bidirectional ELM Method

The basic idea of the method is to find some better hidden nodes parameters $(\mathbf{a}, b)$ which makes the residual error of neural network reduce as quickly as possible. When number of hidden nodes $L \in \{2n+1, n \in \mathbf{Z}\}$, the hidden node parameters $\mathbf{a}, b$ are generated randomly, the same as I-ELM. However, when the number of hidden nodes $L \in \{2n, n \in \mathbf{Z}\}$, the parameters of hidden node $\mathbf{a}, b$ are obtained by Theorem 2. Fig. 2 illustrates the block diagram of the proposed strategy.

B. Bidirectional ELM Method

Theorem 1: Given SLFNs with any bounded nonconstant piecewise continuous function $\mathbf{H} : \mathbf{R} \to \mathbf{R}$ for additive nodes or sine nodes, for any continuous target function f, randomly generated function sequence $\mathbf{H}_{2n+1}^r$, and obtained error feedback function sequence $\mathbf{H}_{2n}^e, n \in Z$, $\lim_{n\to\infty} \|f - (f_{2n-2} + \mathbf{H}_{2n-1}^r \cdot \beta_{2n-1} + \mathbf{H}_{2n}^e \cdot \beta_{2n})\| = 0$ holds with probability 1 if

$$\mathbf{H}_{2n}^e = e_{2n-1} \cdot (\beta_{2n-1})^{-1} \tag{6}$$

$$\beta_{2n} = \frac{\langle e_{2n-1}, \mathbf{H}_{2n}^e\rangle}{\|\mathbf{H}_{2n}^e\|^2} \tag{7}$$

$$\beta_{2n+1} = \frac{\langle e_{2n}, \mathbf{H}_{2n+1}^r\rangle}{\|\mathbf{H}_{2n+1}^r\|^2}. \tag{8}$$

Proof: We first prove that the sequence $\|e_n\|$ is decreasing and bounded below by zero and it converges. Then we further prove $\lim_{n\to\infty} e_n = 0$.

TABLE I
AVERAGE $|(\bar{\beta}_{2n}/\bar{\beta}_{2n-1})|^2$ AND AVERAGE $(\|\bar{e}_{2n}\|/\|\bar{e}_{2n-1}\|)^2$ OVER 50 TRAILS FOR ALL CASES

Datasets	$\left\|\left(\frac{\bar{\beta}_2}{\bar{\beta}_1}\right)\right\|^2$	$\left\|\left(\frac{\bar{\beta}_4}{\bar{\beta}_3}\right)\right\|^2$	$\left\|\left(\frac{\bar{\beta}_6}{\bar{\beta}_5}\right)\right\|^2$	$\left\|\left(\frac{\bar{\beta}_8}{\bar{\beta}_7}\right)\right\|^2$	$\left\|\left(\frac{\bar{\beta}_{10}}{\bar{\beta}_9}\right)\right\|^2$	$\left(\frac{\|\bar{e}_2\|}{\|e_1\|}\right)^2$	$\left(\frac{\|\bar{e}_4\|}{\|e_3\|}\right)^2$	$\left(\frac{\|\bar{e}_6\|}{\|e_5\|}\right)^2$	$\left(\frac{\|\bar{e}_8\|}{\|e_7\|}\right)^2$	$\left(\frac{\|\bar{e}_{10}\|}{\|e_9\|}\right)^2$	S_1	S_2	S_3	S_4	S_5
Fried	0.9465	0.1600	0.1060	0.0603	0.0222	0.0877	0.9934	0.9967	0.9987	0.9992	1.0342	1.1534	1.1027	1.0590	1.0214
Machine CPU	0.6228	0.5048	0.1390	0.0918	0.0744	0.3740	0.5242	0.9099	0.9278	0.9260	0.9968	1.0290	1.0489	1.0196	1.0004
Auto MPG	0.7596	0.3430	0.0331	0.0509	0.0231	0.1748	0.6679	0.9392	0.9688	0.9573	0.9344	1.0109	0.9723	1.0197	0.9804
Wine	0.8656	0.0902	0.0247	0.0164	0.0029	0.1649	0.9620	0.9964	0.9980	0.9992	1.0305	1.0522	1.0211	1.0144	1.0021
Servo	0.6050	0.0776	0.0458	0.0443	0.0140	0.4358	0.9086	0.9569	0.9710	0.9835	1.0410	0.9861	1.0024	1.0159	0.9970

$S_i = \left|\left(\frac{\bar{\beta}_{i+1}}{\bar{\beta}_i}\right)|^2 + \left(\frac{\|\bar{e}_{i+1}\|}{\|e_i\|}\right)\right|^2.$

1) Let $\mathbf{H}_L = \mathbf{H}_1(\mathbf{a}_1, \ldots, \mathbf{a}_L, b_1, \ldots, b_L, \mathbf{x}_1, \ldots, \mathbf{x}_N)$ denotes the hidden-layer output matrix of the SLFN with L hidden node $\{(\mathbf{a}_1, \ldots, \mathbf{a}_L, b_1, \ldots, b_L)\}$. We have $e_{2n-1} = f - [\mathbf{H}_1^r, \mathbf{H}_2^e, \ldots, \mathbf{H}_{2n-1}^r] \cdot [\beta_1, \ldots, \beta_{2n-1}]^T$. Let $\triangle = \|e_{2n-1}\|^2 - \|e_{2n}\|^2$, since $\|e_{2n}\| = \|e_{2n-1} - \mathbf{H}_{2n}^e \cdot \beta_{2n}\|$, we have

$$\begin{aligned}\triangle &= \|e_{2n-1}\|^2 - \|e_{2n-1} - \mathbf{H}_{2n}^e \cdot \beta_{2n}\|^2 \\ &= 2\beta_{2n}\langle e_{2n-1}, \mathbf{H}_{2n}^e\rangle - \|\mathbf{H}_{2n}^e\|^2 \cdot \beta_{2n}^2 \\ &= \|\mathbf{H}_{2n}^e\|^2\left(\frac{2\beta_{2n}\langle e_{2n-1}, \mathbf{H}_{2n}^e\rangle}{\|\mathbf{H}_{2n}^e\|^2} - \beta_{2n}^2\right).\end{aligned} \tag{9}$$

According to (7), we have

$$\begin{aligned}\triangle &= \|\mathbf{H}_{2n}^e\|^2(2\beta_{2n} \cdot \beta_{2n} - \beta_{2n}^2) \\ &= \|\mathbf{H}_{2n}^e\|^2 \cdot \beta_{2n}^2 \geq 0.\end{aligned} \tag{10}$$

In [8, p. 882], Huang *et al.* have proved that $\|e_{2n}\|^2 - \|e_{2n} - \mathbf{H}_{2n+1}^r\beta_{2n+1}\|^2 \geq 0$, thus $\|e_{2n}\| \geq \|e_{2n+1}\|$. Now we have $\|e_{2n-1}\| \geq \|e_{2n}\| \geq \|e_{2n+1}\|$, so the sequence $\{\|e_n\|\}$ is decreasing and bounded below by zero and it $\{\|e_n\|\}$ converges.

2) According to Lemma 1, when the function sequence $\mathbf{H}_n^r$ randomly generated based on any continuous sampling distribution, $\lim_{n\to\infty} \|f - (f_{n-1} + \mathbf{H}_n^r\beta_n)\| = 0$ holds with probability 1 if $\beta_n = \langle e_{n-1}, \mathbf{H}_n^r\rangle / \|\mathbf{H}_n^r\|^2$. This means $\mathbf{H}$ does not affect the universal approximation capability of SLFNs, and the SLFNs f_n can converge to any continuous target function f by only adjusting the output weights $\beta_n = \langle e_{n-1}, \mathbf{H}_n^r\rangle / \|\mathbf{H}_n^r\|^2$. Thus if we set

$$\mathbf{H}_{2n}^r \equiv \mathbf{H}_{2n}^e = e_{2n-1} \cdot (\beta_{2n-1})^{-1} \tag{11}$$

we get

$$\begin{aligned}&\lim_{n\to\infty} \|f - (\mathbf{H}_1^r\beta_1 + \cdots + \mathbf{H}_{2n-1}^r\beta_{2n-1} + \mathbf{H}_{2n}^e\beta_{2n})\| \\ &= \lim_{n\to\infty} \|f - (\mathbf{H}_1^r\beta_1 + \cdots + \mathbf{H}_{2n-1}^r\beta_{2n-1} + \mathbf{H}_{2n}^r\beta_{2n})\| \\ &= \lim_{n\to\infty} \|f - (f_{n-1} + \mathbf{H}_n^r\beta_n)\| = 0.\end{aligned} \tag{12}$$

■

Theorem 2: Given a sigmoid or sine activation function $h : \mathbf{R} \to \mathbf{R}$. Given an error feedback function sequence $\mathbf{H}_{2n}^e(\mathbf{x}, \mathbf{a}, b)$, according to Theorem 1

$$\mathbf{H}_{2n}^e = e_{2n-1} \cdot (\beta_{2n-1})^{-1}. \tag{13}$$

If the activation function h is sin/cos, given a normalized function $u : \mathbf{R} \to [0, 1]$, If activation function h is sigmoid, given a normalized function $u : \mathbf{R} \to (0, 1]$. Then for any continuous target function f, the randomly generated function sequence $\mathbf{H}_{2n+1}^r$, $\lim_{n\to\infty} \|f - (\mathbf{H}_1^r \cdot \beta_1 + \hat{\mathbf{H}}_2^e(\hat{\mathbf{a}}_2, \hat{b}_2) \cdot \beta_2 + \cdots + \mathbf{H}_{2n-1}^r \cdot \beta_{2n-1} + \hat{\mathbf{H}}_{2n}^e(\hat{\mathbf{a}}_{2n}, \hat{b}_{2n}) \cdot \beta_{2n}\| = 0$ holds with probability 1 if

$$\begin{aligned}\hat{\mathbf{a}}_{2n} &= h^{-1}(u(\mathbf{H}_{2n}^e)) \cdot \mathbf{x}^{-1} \\ \hat{b}_{2n} &= \sqrt{\text{mse}(h^{-1}(u(\mathbf{H}_{2n}^e)) - \hat{\mathbf{a}}_{2n} \cdot \mathbf{x})} \end{aligned} \tag{14}$$

$$\hat{\mathbf{H}}_{2n}^e = u^{-1}(h(\hat{\mathbf{a}}_{2n} \cdot \mathbf{x} + \hat{b}_{2n})) \tag{15}$$

where h^{-1} and u^{-1} represent its reverse functions, respectively, if h is a sine activation function, $h^{-1}(\cdot) = arcsin(\cdot)$ if h is a sigmoid activation function, $h^{-1}(\cdot) = -\log(1/(\cdot) - 1)$.

Proof: The validity of this theorem is obvious from Theorem 1 and Lemma 1. Similar to Theorem 1(b), if set $\mathbf{H}_{2n}^r = \hat{\mathbf{H}}_{2n}^e$, we have $\lim_{n\to\infty} \|f - (f_{2n-1} + \hat{\mathbf{H}}_{2n}^e\beta_{2n})\| = 0$. ■

Remark 1: Different from other incremental ELM learning methods in which the hidden-layer input weights $\mathbf{a}, b$ are generated randomly, in proposed B-ELM, when hidden-node numbers $L \in 2n$, sequence $\mathbf{H}_{2n}^e$ is generated by (7). It should be noted that, by doing so, Theorem 1 greatly enhances the convergence rate of ELM (see Figs. 3–8). According to (6), (10) can be rewritten as

$$\|e_{2n-1}^2\| - \|e_{2n}^2\| = \|\mathbf{H}_{2n}^e\|^2 \cdot \beta_{2n}^2 = \left(\frac{\beta_{2n}}{\beta_{2n-1}}\right)^2 \cdot \|e_{2n-1}\|^2. \tag{16}$$

Thus, we have

$$\frac{\beta_{2n}^2}{\beta_{2n-1}^2} + \frac{\|e_{2n}\|^2}{\|e_{2n-1}\|^2} = 1. \tag{17}$$

Equation (17) shows an important relationship between e and β and we have named this equation the error-output weight ellipse. Based on this equation, we have three results.

1) If β_{2n} and H_{2n}^e are generated by (7) and (6), $|\beta_{2n}| \leq |\beta_{2n-1}|$ and $\|e_{2n}\| \leq \|e_{2n-1}\|$.

2) The universal approximation capability of SLFNs only depends on parameter β of SLFNs, not on $\mathbf{a}, b$ at all. This

result is as the same as that of [8]. However, in this brief, we prove this result by giving a very simple expression.

3) The learning effectiveness of B-ELM only depends on $|\beta_{2n}|/|\beta_{2n-1}|$, because if $|\beta_{2n}|/|\beta_{2n-1}| \to 1$, $\|e_{2n}\|/\|e_{2n-1}\| \to 0$ and, then, $\|e_{2n}\| << \|e_{2n-1}\|$.

C. Learning the Effectiveness of B-ELM

In this Section, we further analyze the learning effectiveness of B-ELM because, in real applications, we find that B-ELM with only two or four hidden nodes can achieve a similar generalization performance as I-ELM with hundreds of hidden nodes (see Figs. 3–8). We want to prove why B-ELM can provide good generalization performance at an extremely early learning stage (with very few hidden nodes).

Lemma 2: For any continuous target function f, if $T = f$[1] and SLFNs is trained by B-ELM: randomly generated function sequence $\mathbf{H}^r_{2n+1}$, obtained error feedback function sequence $\mathbf{H}^e_{2n}$ and $\beta_n = E_{n-1} \cdot \mathbf{H}_n^T/\mathbf{H}_n \cdot \mathbf{H}_n^T$, we have following results:

$$\|e_{2n}\| = \sqrt{\tau_{2n-1}} \cdot \|e_{2n-1}\| \tag{18}$$

$$\lim_{n\to+\infty} \tau_n = 1, n \in \mathbf{Z} \tag{19}$$

where $\tau_{2n-1} = \|E_{2n-1}\| \cdot \|(\mathbf{H}^e_{2n})^T\|/\|\mathbf{H}^e_{2n}\| \cdot \|(\mathbf{H}^e_{2n})^T\| - \|E_{2n-1} \cdot (\mathbf{H}^e_{2n})^T/\mathbf{H}^e_{2n} \cdot (\mathbf{H}^e_{2n})^T\|$.

Proof: We first prove (18). Then we further prove $\lim_{n\to+\infty} \tau_n = 1$.

1) Since $\beta_n \in \mathbf{R}$, $\mathbf{H}^e_{2n} = e_{2n-1} \cdot (\beta_{2n-1})^{-1}$, and $f = T$, we get

$$\begin{aligned}|\beta_{2n}| = \|\beta_{2n}\| &= \left\| \frac{E_{2n-1} \cdot (\mathbf{H}^e_{2n})^T}{\mathbf{H}^e_{2n} \cdot (\mathbf{H}^e_{2n})^T} \right\| \\ &\le \frac{\|E_{2n-1}\| \cdot \|(\mathbf{H}^e_{2n})^T\|}{\|\mathbf{H}^e_{2n}\| \cdot \|(\mathbf{H}^e_{2n})^T\|} = \frac{\|E_{2n-1}\|}{\|\mathbf{H}^e_{2n}\|} \\ &= \|\beta_{2n-1}.\|.\end{aligned} \tag{20}$$

Thus, if set $\tau_{2n-1} = |\beta_{2n-1}| - |\beta_{2n}| = \|E_{2n-1}\| \cdot \|(\mathbf{H}^e_{2n})^T\| \, \|\mathbf{H}^e_{2n}\| \cdot \|/(\mathbf{H}^e_{2n})^T\| - \|E_{2n-1} \cdot (\mathbf{H}^e_{2n})^T \mathbf{H}^e_{2n} \cdot /(\mathbf{H}^e_{2n})^T\|$, we have

$$\frac{\|\beta_{2n}\|}{\|\beta_{2n-1}|} = \frac{|\beta_{2n-1}| - \tau_{2n-1}}{|\beta_{2n-1}|}. \tag{21}$$

Considering (17), we get

$$\|e_{2n-1}\| - \|e_{2n}\| = \|e_{2n-1}\| \cdot (1 - \sqrt{\tau}). \tag{22}$$

2) In [8, p. 883], it is proved that $\|e_{2n-1} - e_{2n}\| = \|e_{2n-1}\| - \|e_{2n}\|$ and $\lim_{n\to+\infty}(\|e_{2n-1}\| - \|e_{2n}\|) = \lim_{n\to+\infty} \|e_{2n-1} - e_{2n}\| = 0$. Further, in [8, p. 883] it is proved that $\lim_{n\to+\infty} \|e_{2n-1}\| = \gamma > 0$. Based on these two results and according to (22), we have $\lim_{n\to+\infty} \sqrt{\tau_n} = 1$. ■

Theorem 3: For any continuous target f, if SLFNs is trained by B-ELM, and if $f = T$, we have $\{(\tau_{2n-1}, |\beta_{2n}|/|\beta_{2n-1}|)|\tau_{2n-1} \in \{0, 1\}, |\beta_{2n}|/|\beta_{2n-1}| \in \{1, 0\}$.

[1]In practice, β_n can only be calculated when the exact functional form of e_{n-1} is available, which is obviously impossible as the true f is unknown [8] Similar to other learning methods [1]–[13], we set $f = T$. Hence, we can get $\beta_n = E_{n-1} \cdot \mathbf{H}_n^T/\mathbf{H}_n \cdot \mathbf{H}_n^T$ and $E_n = e_n$. From the application point of view, the equation $f = T$ is obviously valid because the true aim of training network is make the network output approximate the given target output samples T, not unknown target function f.

Algorithm 1 B-ELM Algorithm

```
Given a training set $\{(\mathbf{x}_i, t_i)\}_{i=1}^N \subset \mathbf{R}^n \times \mathbf{R}$, the hidden-node
output function $\mathbf{H}(\mathbf{a}, b, \mathbf{x})$, the continuous target function $f$
and the hidden-node number $L$, and the expected learning
accuracy $\eta$:
Step 1) **Initialization**: Let number of hidden nodes $L = 0$,
residual error $E = T$, where $t = [t_1, \ldots, t_N]$.
Step 2) **Learning step**:
while $L < L_{max}$, $\|E\| > \eta$ do
  increase by one the number of hidden nodes $L : L = L+1$
  if $L \in \{2n+1, n \in \mathbf{Z}\}$ then
    a) assign random input weight $\mathbf{a}_L$ and bias $b_L$ for new
    hidden node $L$;
    b) calculate the output weight $\beta_L$ for the new hidden
    node according to (8);
    c) calculate $E$ after adding the new hidden node $L$:$E =
    E - \mathbf{H}_L \cdot \beta_L$
  end if
  if $L \in \{2n, n \in \mathbf{Z}\}$ then
    a) calculate the error feedback function sequence $\mathbf{H}_L$
    according to (6);
    b) calculate the input weight $\mathbf{a}_L$, bias $b_L$ and update $\mathbf{H}_L$
    for the new hidden node $L$ based on (14) and (15);
    c) calculate the output weight $\beta_L$ for the new hidden
    node according to (7);
    d) calculate $E$ after adding the new hidden node $L$:$E =
    E - \mathbf{H}_L \cdot \beta_L$
  end if
end while
```

Proof: Based on (21), we get

$$|\beta_{2n}| = |\beta_{2n-1}| - \tau_{2n-1} \cdot |\beta_{2n-1}|. \tag{23}$$

According to the definition of τ, we have $|\beta_{2n}| = |\beta_{2n-1}| - \tau_{2n-1}$. Thus the value of τ_{2n-1}, β_{2n-1}, and β_{2n} should satisfy the following equation:

$$|\beta_{2n}| = |\beta_{2n-1}| - \tau_{2n-1} = |\beta_{2n-1}| - \tau_{2n-1} \cdot |\beta_{2n-1}|. \tag{24}$$

In order for this equation to be valid, we should have the following.

1) When $\tau_{2n-1} = 0$, based on (17) and (18), we have $|\beta_{2n}|/|\beta_{2n-1}| = 1$.

2) When $\tau_{2n-1} \neq 0$, $|\beta_{2n-1}|$ must equal to 1. Based on this result, we get $|\beta_{2n}| = 1 - \tau_{2n-1}$.

Considering (17), we get

$$1 = \frac{\beta^2_{2n}}{\beta^2_{2n-1}} + \frac{\|e_{2n}\|^2}{\|e_{2n-1}\|^2} = 1 - \tau_{2n-1} + (\tau_{2n-1})^2. \tag{25}$$

We get the solutions $\tau_{2n-1} = 1$ and $|\beta_{2n}| = 0$. Thus when $\tau_{2n-1} \neq 0$, τ_{2n-1} is only equal to 1 and $|\beta_{2n}|/|\beta_{2n-1}|$ is only equal to 0. ■

Remark 2: Under the condition of $f = T$, Theorem 3 indicates a very interesting result. When an SLFN with two

TABLE II
PERFORMANCE COMPARISON (MEAN-MEAN TESTING RMSE; TIME-TRAINING TIME)

Datasets	B-ELM (2 nodes)		I-ELM (2 nodes)		EM-ELM (2 nodes)		EI-ELM (2 nodes)	
	Sine		Sine		Sine		Sine, $p = 50$	
	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)
Machine CPU	**0.0794**	0.0053	0.1628	0.0036	0.1009	0.0024	0.0890	0.1259
Concrete data	**0.1554**	0.0019	0.4815	0.0028	0.2293	0.0010	0.1613	0.1563
Wine quality	**0.1310**	0.0050	0.3703	0.0048	0.1515	0.0027	0.1441	0.3246
Bank	**0.1583**	0.0061	0.2924	0.0064	0.2061	0.0031	0.1776	0.2672
Fried	**0.1283**	0.0057	0.5429	0.0129	0.3399	0.0015	0.2214	1.7288
Auto MPG	0.1447	0.0004	0.4628	0.0004	0.1768	0.0001	**0.1223**	0.1289
Concrete Slump	**0.3933**	0.0004	0.6802	0.0008	0.5212	0.0006	0.4219	0.0690

TABLE III
PERFORMANCE COMPARISON (MEAN-MEAN TESTING RMSE; TIME-TRAINING TIME)

Datasets	B-ELM (10 nodes)		I-ELM (10 nodes)		EM-ELM (10 nodes)		EI-ELM (10 nodes)	
	Sine		Sine		Sine		Sine, $p = 50$	
	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)
Machine CPU	**0.0561**	0.0103	0.1559	0.0069	0.0663	0.0056	0.0695	0.3331
Concrete data	0.1519	0.0072	0.4414	0.0094	0.1689	0.0102	0.1486	0.4150
Wine quality	0.1335	0.0128	0.2671	0.0271	0.1325	0.0181	0.1331	0.7248
Bank	**0.1188**	0.0977	0.1940	0.0104	0.1815	0.0128	0.1385	0.6517
Fried	**0.1223**	0.0130	0.2182	0.0139	0.1408	0.0005	0.1272	1.9766
Servo	0.1971	0.0101	0.2889	0.0087	0.1543	0.1417	**0.1528**	0.2866
Concrete slump	**0.3374**	0.0022	0.6791	0.0034	0.5055	0.0031	0.3501	0.1462

hidden nodes is trained by B-ELM, it is obvious [2] that $0 < \|e_1\| - \|e_2\|, \|e_1\| \neq 0$. So we get $\|e_2\| \|e_1\| \neq 1$. Based on Theorem 3, we get $\|e_2\|/\|e_1\| = 0$. According to remark 1, we get $e_2 = 0$. Thus, in theory, B-ELM with two hidden nodes can reduce network output error e to 0.

In real applications, because $\mathbf{H}^e \approx \hat{\mathbf{H}}^e$(see Theorem 2), we can find from Table I that $S \approx 1$, $|\bar{\beta}_{2n}/\bar{\beta}_{2n-1}|$ is approximately equal to 1 or 0, and $(\|\bar{e}_{2n}\|/\|\bar{e}_{2n-1}\|)^2$ is approximately equal to 0 or 1.

D. Pseudo-Code for B-ELM Method

The proposed B-ELM for SLFN can be summarized as in Algorithm 1.

IV. EXPERIMENTAL VERIFICATION

To examine the performance of our proposed algorithm (B-ELM), in this section, we test them on some benchmark applications. The simulations are conducted in MATLAB 2009a running on the same Windows 7 machine with at 2 GB of memory and an i5-430 (2.33 GHZ) processor. Neural networks are tested in EM-ELM, I-ELM, EI-ELM, and the proposed B-ELM.

Nine different datasets were chosen for the experiments. All datasets were preprocessed in the same way. Ten different random permutations of the whole dataset were taken without replacement, and half of them were used to create the training set and the remaining half for the test set. Then the input data were normalized into $[-1, 1]$ while the output data for regression were normalized into the range [0, 1]. The average results were obtained over 50 trials for all problems.

B-ELM, I-ELM, EM-ELM, and EI-ELM were first compared in three real benchmark problems: Machine CPU, Fried, and Bank. In these cases, all the algorithms increase the hidden nodes one by one. Figs. 3–8 show the RMSE averaged over 50 experiments obtained by these methods with sine type and sigmoid type hidden nodes. It can be seen that proposed B-ELM can obtain much better generalization performance than other methods when a sigmoid/sine hidden node is used. More importantly, we find that the testing RMSE obtained by B-ELM is reduced to a very small value when only two or four hidden nodes are used. As mentioned in [12], in real applications, the network growing procedure could stop if the network output error reduced very slowly, which means the RMSE of B-ELM can reach limiting and small value at the extremely early learning stage.

Further comparisons were conducted on nine real benchmark regression problems shown in Tables II–V. In these tables, the close results obtained by different algorithms are underlined and the apparently better results are shown in boldface. B-ELM, I-ELM, and EI-ELM increase the hidden nodes one by one. However, in the implementation of EM-ELM, the initial SLFNs are given two hidden nodes and then new hidden nodes are added one by one. As observed from Tables II–V and Figs. 3–8, the advantage of the B-ELM on testing RMSE is quite

[2]In [12, p. 1356], Huang *et al.* indicate "the network growing procedure could stop if the network output error reduced very slowly, say $|e_{n+1} - e_n| < \eta$. Here we do not prove $\|e_1\| - \|e_2\| \neq 0$, but all the ELM simulation results [1]–[7] and [9]–[13] show $\|e_1\| - \|e_2\| \gneqq 0$.

TABLE IV
PERFORMANCE COMPARISON (MEAN-MEAN TESTING RMSE; TIME-TRAINING TIME)

Datasets	B-ELM (2 nodes)		I-ELM (2 nodes)		EM-ELM (2 nodes)		EI-ELM (2 nodes)	
	Sigmoid		Sigmoid		Sigmoid		Sigmoid, $p = 50$	
	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)
Machine CPU	**0.0520**	0.0016	0.1261	0.0019	0.0719	0.0001	0.0653	0.1291
Concrete data	**0.1391**	0.0048	0.2271	0.0037	0.1758	0.0003	0.1634	0.1699
Wine quality	**0.1268**	0.0055	0.1639	0.0070	0.1353	0.0004	0.1343	0.3604
Bank	**0.0521**	0.0033	0.2850	0.0115	0.1739	0.0007	0.1659	0.3012
Fried	**0.1280**	0.0426	0.5660	0.0544	0.2410	0.1154	0.1870	3.9127
Servo	0.1871	0.0014	0.3143	0.1417	0.1739	0.0005	0.1869	0.1273
Concrete slump	**0.2883**	0.0001	0.5527	0.0017	0.5017	0.0001	0.5125	0.0738
Puma	**0.4017**	0.0048	0.6680	0.0112	0.4886	0.0001	0.4784	0.5389
Auto MPG	**0.0915**	0.0033	0.2118	0.0025	0.1092	0.0001	0.1076	0.1253

TABLE V
PERFORMANCE COMPARISON (MEAN-MEAN TESTING RMSE; TIME-TRAINING TIME)

Datasets	B-ELM (10 nodes)		I-ELM (10 nodes)		EM-ELM (10 nodes)		EI-ELM (10 nodes)	
	Sigmoid		Sigmoid		Sigmoid		Sigmoid, $p = 50$	
	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)	Mean	Time (s)
Machine CPU	0.0522	0.0078	0.1257	0.0089	0.0566	0.0070	0.0571	0.3638
Concrete data	0.1325	0.0125	0.2281	0.0098	0.1372	0.0114	0.1467	0.4098
Wine quality	0.1261	0.0128	0.1638	0.0197	0.1277	0.0157	0.1313	0.8159
Bank	**0.0530**	0.0098	0.1906	0.0296	0.1047	0.0106	0.1463	0.7134
Fried	**0.0813**	0.0432	0.2159	0.0604	0.0977	0.1206	0.1008	4.0767
Servo	0.1971	0.0101	0.2889	0.0087	0.1543	0.1417	0.1528	0.2866
Concrete slump	**0.2562**	0.0039	0.5399	0.0036	0.3724	0.0030	0.4037	0.1560

TABLE VI
PERFORMANCE COMPARISON BETWEEN B-ELM AND OTHER ELM METHODS. NEW HIDDEN NODES ARE ADDED ONE BY ONE. ALL THE METHODS STOP GROWING WHEN THE EXPECTED ACCURACY IS REACHED

Datasets (stop RMSE)	B-ELM			I-ELM			EM-ELM			EI-ELM		
	Sigmoid			Sigmoid			Sigmoid			Sigmoid, $p = 50$		
	Node	Time (s)	Testing RMSE	Node	Time (s)	Testing RMSE	Node	Time (s)	Testing RMSE	Node	Time (s)	Testing RMSE
Bank (0.065)	**6.7**	**0.0094**	0.0644	909.4	1.6388	0.0652	19.8	0.0421	0.0643	103.8	7.1901	0.0652
Fried (0.088)	**2.0**	**0.1654**	0.0859	410.1	4.1028	0.0880	14.1	0.2691	0.0864	42.4	19.6210	0.0870
Machine CPU (0.060)	**2.0**	**0.0001**	0.0517	85.4	0.0975	0.0612	6.8	0.0016	0.0583	6.6	0.2426	0.0593
Wine(0.130)	**2.0**	**0.0047**	0.1273	380.0	0.8112	0.1319	6.9	0.0109	0.1296	14.9	1.2558	0.1306
Concrete (0.138)	**2.6**	**0.0062**	0.1373	290.7	0.3393	0.1396	8.5	0.0570	0.1355	21.0	0.8346	0.1392
Concrete slump (0.255)	**2.1**	**0.0382**	0.2878	293.4	0.1786	0.4082	10.1	0.7800	0.3968	107.4	1.4914	0.3671
Auto MPG (0.09)	**7.9**	**0.0057**	0.0960	740.2	0.8003	0.0939	8.1	0.0123	0.0923	86.7	2.9531	0.0933

obvious. In Table II, for the Fried problem, the B-ELM runs 300 times faster than the EI-ELM and the testing RMSE of EI-ELM is 2 times larger than that of B-ELM. The B-ELM runs 1.5 times faster than the I-ELM and the testing RMSE for the obtained I-ELM is 5 times larger than that of B-ELM. In real applications, SLFNs with two hidden nodes are extremely small network structures, meaning that, after training, this small sized network may respond to new external unknown stimuli much faster and much more accurately than other incremental ELM algorithms in real deployment.

In order to demonstrate the advantage of the B-ELM on the network structure, the hidden-node numbers between B-ELM and other algorithms were also calculated in these problems. For these problems, Tables VI and VII display the performance evaluation between B-ELM and other ELM methods. As seen from the simulation results given in these tables, B-ELM is faster than other traditional ELMs. Interestingly, the hidden-node numbers obtained by our proposed new algorithm B-ELM are much smaller than those of the other methods. In Table VI, for Bank data, the B-ELM learning algorithm spent 0.0094 s CPU time with 6.7 hidden nodes; however, it takes 1.6388, 0.0821, and 7.1901 s for I-ELM, EM-ELM, and EI-ELM to reach the same training error 0.0650. The new B-ELM runs 170, 10,

TABLE VII

PERFORMANCE COMPARISON BETWEEN B-ELM AND OTHER ELM METHODS NEW HIDDEN NODES ARE ADDED ONE BY ONE. ALL THE METHODS STOP GROWING WHEN THE EXPECTED ACCURACY IS REACHED

Datasets (stop RMSE)	B-ELM			I-ELM			EM-ELM			EI-ELM		
	Sin			Sin			Sin			Sin, $p = 50$		
	Node	Time (s)	Testing RMSE	Node	Time (s)	Testing RMSE	Node	Time (s)	Testing RMSE	Node	Time (s)	Testing RMSE
Machine CPU (0.060)	**5.3**	0.0025	0.0760	81.1	0.1154	0.0757	7.6	0.0031	0.0800	8.8	0.3916	0.0800
Wine(0.135)	**2.0**	**0.0071**	0.1366	233.7	0.6318	0.1353	8.0	0.0150	0.1335	6.8	0.7394	0.1326
Concrete (0.138)	**2.6**	**0.0042**	0.1373	290.7	0.3393	0.1396	8.5	0.0070	0.1355	21.0	0.8346	0.1392
Concrete slump (0.255)	**2.1**	**0.0382**	0.2878	293.4	0.1786	0.4082	10.1	0.7800	0.3968	107.4	1.4914	0.3671
Auto MPG (0.09)	7.9	0.0057	0.0960	740.2	0.8003	0.0939	8.1	**0.0023**	0.0923	86.7	2.9531	0.0933
Fried (0.12)	**6.4**	**0.0110**	0.1160	972	7.6472	0.1308	50.1	3.5833	0.1188	78.7	28.5435	0.1199

and 800 times faster than the I-ELM, EM-ELM, and EI-ELM respectively. Meanwhile, the optimal hidden-node numbers for the obtained I-ELM, EM-ELM, and EI-ELM is 3, 15, and 134 times larger than the other optimal hidden-node numbers for B-ELM, respectively. Similar results can be also obtained for the rest of the cases. That means that, after training and deployment, the B-ELM may react to new observations much faster than the other methods in a real application.

Huang *et al.* [2], [8], and [12] have systematically investigated the performance of I-ELM, EM-ELM, SVR, and BP for most datasets tested in this brief. It is found that ELM and I-ELM obtain the same generalization performance as SVR but in much simpler and faster way. This is to say, the network size of SVR is higher than that of I-ELM. As observed from our testing results, the proposed B-ELM obtains a more compact network architecture than EM-ELM and I-ELM. Thus, the network size of B-ELM is much lower than SVR. Furthermore, it also found in [12] that EM-ELM usually obtains better generalization performance than BP under the same number of hidden nodes. Interestingly, from our testing results, the proposed B-ELM obtains better performance than EM-ELM and I-ELM in the terms of learning speed and network structure. Thus, the proposed B-ELM performs better than BP as well.

V. CONCLUSION

In this brief, a new learning algorithm called bidirectional extreme learning machine (B-ELM) was presented. Unlike other incremental ELM methods, in this new approach some hidden node is incrementally updated efficiently during the growth of the networks. It was proven in theory that B-ELM can greatly enhance the learning effectiveness, reduce the number of hidden nodes, and eventually further increase the learning speed. The simulation results on sigmoid/sine type of hidden nodes showed that, compared to other incremental ELM methods, the new approach could significantly reduce the NN training time by hundreds of times. The performance of our method on other types of hidden nodes will be reported in the future.

ACKNOWLEDGMENT

The authors would like to thank the editor and anonymous reviewers for their invaluable suggestions, which improved the quality of this brief dramatically. They would like to thank G.-B. Huang, Nanyang Technological University, Singapore, for constructive suggestions.

REFERENCES

[1] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Real-time learning capability of neural networks," School Electr. Electron. Eng., Nanyang Technological Univ., Singapore, Tech. Rep. ICIS/45/2003, 2003.

[2] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[3] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 2. Budapest, Hungary, Jul. 2004, pp. 985–990.

[4] T. Y. Kwok and D. Y. Yeung, "Objective functions for training new hidden units in constructive neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1131–1148, Sep. 1997.

[5] Y. Lan, Y. C. Soh, and G. B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, nos. 16–18, pp. 3028–3038, 2010.

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

[7] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

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

[9] G. B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3460–3468, 2008.

[10] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.

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

[12] G. Feng, G. B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1352–1357, Aug. 2009.

[13] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern., Part B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.