Neural Comput & Applic
DOI 10.1007/s00521-014-1567-3

Orthogonal incremental extreme learning machine for regression and multiclass classification

Li Ying

Received: 10 September 2013 / Accepted: 11 March 2014

Abstract Single-hidden-layer feedforward networks with randomly generated additive or radial basis function hidden nodes have been theoretically proved that they can approximate any continuous function. Meanwhile, an incremental algorithm referred to as incremental extreme learning machine (I-ELM) was proposed which outperforms many popular learning algorithms. However, I-ELM may produce redundant nodes which increase the network architecture complexity and reduce the convergence rate of I-ELM. Moreover, the output weight vector obtained by I-ELM is not the least squares solution of equation $\boldsymbol{H\beta} = \boldsymbol{T}$. In order to settle these problems, this paper proposes an orthogonal incremental extreme learning machine (OI-ELM) and gives the rigorous proofs in theory. OI-ELM avoids redundant nodes and obtains the least squares solution of equation $\boldsymbol{H\beta} = \boldsymbol{T}$ through incorporating the Gram–Schmidt orthogonalization method into I-ELM. Simulation results on nonlinear dynamic system identification and some benchmark real-world problems verify that OI-ELM learns much faster and obtains much more compact neural networks than ELM, I-ELM, convex I-ELM and enhanced I-ELM while keeping competitive performance.

Keywords Feedforward neural networks · Incremental extreme learning machine · Least squares solution · Gram–Schmidt orthogonalization method · Convergence rate

L. Ying (✉)
School of Science, Qi Lu University of Technology, Jinan 250353, Shandong, China
e-mail: lzhbb07@sina.com

1 Introduction

As a specific type of feedforward neural networks (FNNs), the single-hidden-layer feedforward network (SLFN) plays an important role in practical applications and has been investigated extensively in both theory and application in the past decades [1–5]. It has been proved that an SLFN with at most N hidden nodes can learn any N arbitrary distinct samples with zero error when all the parameters of the network are allowed to be tuned freely [1]. However, tuning all the parameters of the network may cause learning complicated and inefficient. In [6], Huang et al. have demonstrated that an SLFN with no more than N random hidden nodes and infinitely differentiable activation functions can learn any N arbitrary distinct samples with any accuracy. Meanwhile, extreme learning machine (ELM) which randomly generates the parameters of the hidden nodes and analytically obtains the weights of the output layer was proposed for SLFNs. Moreover, it has been proved in theory that ELM has less computational complexity and better generalization performance than conventional gradient-descent algorithms. As a result, it has attracted many researchers' attention and large numbers of good results have emerged [7–19].

Recently, Huang et al. [12] have rigorously proved that SLFNs with randomly generated additive or RBF hidden nodes can work as universal approximators and proposed an algorithm called incremental extreme learning machine (I-ELM). I-ELM adds random hidden nodes to the existent network one by one and freezes all the parameters of the existing nodes when a new random hidden node is added. The output weights of the new added node are analytically calculated by a simple formula. I-ELM is fully automatic in the sense that there are no control parameters needed to be manually set by users except for target errors and the

Published online: 27 March 2014

allowed maximum number of hidden nodes. Moreover, I-ELM performs well in regression applications at high learning speed. Although the merits of the I-ELM are obvious, there still exist some issues to be settled:

1. I-ELM may generate redundant nodes which play a very minor effect on the outputs of the SLFN but increase the complexity of the network architecture.
2. I-ELM needs more hidden nodes than ELM because of the slower convergence rate. The number of hidden nodes in I-ELM is sometimes larger than that of the training samples.
3. The solution of $\boldsymbol{H\beta} = \boldsymbol{T}$ obtained by I-ELM is not the least squares solution indicating the solution is not optimal.
4. I-ELM is rarely used to solve multiclass classification problems.

Therefore, many improved I-ELM algorithms were proposed to deal with the issues in succession. Convex incremental extreme learning machine (CI-ELM) was proposed by incorporating the Barron's convex optimization learning method into I-ELM [13]. The main difference between I-ELM and CI-ELM is that CI-ELM recalculates the output weights of the existing hidden nodes when a new random hidden node is added. CI-ELM obtains faster convergence rate and more compact neural networks than I-ELM while retaining the same simplicity and performance. In [14], an enhanced I-ELM (EI-ELM) was proposed to avoid redundant nodes. The only difference between EI-ELM and I-ELM is that EI-ELM randomly generates several hidden nodes at each training step and adds the one leading to the largest residual error decreasing to the existing SLFN. Generally speaking, EI-ELM aims to obtain more compact neural network than the one obtained by I-ELM. Simulation results on some benchmark real-world regression problems verify that EI-ELM learns faster and obtains much more compact neural network while achieving better performance. However, the issues of I-ELM mentioned above are not thoroughly settled by CI-ELM and EI-ELM.

In this paper, we propose a method called orthogonal extreme learning machine (OI-ELM) to further settle the aforementioned issues of I-ELM and give the rigorous proofs in theory. The proposed method incorporates Gram–Schmidt orthogonalization method into I-ELM and obtains the least squares solution. The proof of the convergence of OI-ELM is then given in this paper. Simulation results on nonlinear dynamic system identification and some benchmark real-world problems verify that OI-ELM can achieve faster convergence rates, much more compact neural network and better generalization performance than both I-ELM and the improved I-ELM algorithms while retaining their simplicity. In a word, the main contributions of this paper are listed as follows:

1. The Gram–Schmidt orthogonalization method is incorporated into I-ELM to propose more effective OI-ELM;
2. Compared with the original I-ELM, OI-ELM can achieve more compact neural networks, faster convergence rate and better generalization performance. Moreover, we give the proof in theory that the hidden nodes of OI-ELM are not more than the training samples;
3. We get the least squares solution of $\boldsymbol{H\beta} = \boldsymbol{T}$ by OI-ELM and give the rigorous proof in theory;
4. We extend OI-ELM to settle both regression and multiclass classification problems.

This paper is organized into five sections. Section 2 briefly reviews I-ELM. The details of the proposed OI-ELM algorithm are described in Sect. 3. Section 4 presents the experiments and simulations results which show the superior performance of the OI-ELM compared with ELM, I-ELM, CI-ELM and EI-ELM. Finally, a summary of this paper is given in Sect. 5.

2 Brief reviews of I-ELM

In this section, we briefly describe the essence of I-ELM to provide the necessary background for OI-ELM. Consider N arbitrary distinct samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in R^n, \boldsymbol{t}_i \in R^m, 1 \le i \le N\}$, where $\boldsymbol{x}_i = (x_{i1}, x_{i2},\ldots,x_{in})^{\mathrm{T}}$ is the input vector, and $\boldsymbol{t}_i = (t_{i1}, t_{i2},\ldots,t_{im})^{\mathrm{T}}$ is the target output vector. The SLFN with L additive hidden nodes and activation function $g(x)$ can be represented by

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^T \boldsymbol{x}_j + b_i\right) = \boldsymbol{o}_j, \quad 1 \le j \le N \tag{1}$$

where $\boldsymbol{w}_i = (w_{i1}, w_{i2},\ldots, w_{in})^{\mathrm{T}}$ is the weight vector linking the input layer with the ith hidden node. b_i is the threshold of the ith hidden node. $\boldsymbol{\beta}_i = (\beta_{i1}, \beta_{i2},\ldots, \beta_{im})^{\mathrm{T}}$ is the weight vector linking the output layer with the ith hidden node. $\boldsymbol{o}_j$ is the output of the SLFN with respect to the input vector $\boldsymbol{x}_j$. The activation function $g(x)$ includes all bounded nonconstant piecewise continuous functions. The SLFN with L hidden nodes can approximate these N samples with zero error when all the parameters are allowed to be adjusted freely, i.e., there exist $\boldsymbol{\beta}_i$, $\boldsymbol{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g\left(\boldsymbol{w}_i^T \boldsymbol{x}_j + b_i\right) = \boldsymbol{t}_j, \quad 1 \le j \le N \tag{2}$$

The above N equations can be compactly rewritten as the matrix equation

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

where

$$\boldsymbol{H} = \begin{bmatrix} g_1(\boldsymbol{x}_1) & \cdots & g_L(\boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ g_1(\boldsymbol{x}_N) & \cdots & g_L(\boldsymbol{x}_N) \end{bmatrix}_{N\times L} \quad \boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L\times m}$$

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N\times m} \tag{4}$$

In Eq. (4), $g_i(\boldsymbol{x}_j) = g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_j + b_i)$, $\boldsymbol{H}$ is called the hidden layer output matrix of the SLFN. The ith column of $\boldsymbol{H}$ is the outputs of the ith hidden node with respect to the N input vectors. The jth row of $\boldsymbol{H}$ is the outputs of all the hidden neurons with respect to the input vector $\boldsymbol{x}_j$.

An active topic on the SLFNs is how to determine the parameters $\boldsymbol{w}_i$, b_i, $\boldsymbol{\beta}_i$ such that the network can approximate the N samples with any minor error [3, 5, 15, 16, 20]. Unlike the conventional neural network technology, Huang et al. proposed I-ELM which randomly adds hidden nodes to the network one by one and freezes all the weights of the existing hidden nodes when a new hidden node is added. The output weights of the new added node are analytically determined by a simple formula. I-ELM is a ''random search'' method in the sense that whose residual error will decrease in theory whenever a new random hidden node is added to the network. I-ELM is an automatic algorithm in the sense that it automatically adds random hidden nodes to the network one by one until achieving the expected training accuracy or reaching the maximum number of hidden nodes. I-ELM algorithm can be summarized as follows.

I-ELM Algorithm [12] Given N arbitrary distinct samples $\{(\boldsymbol{x}_i, t_i)|\boldsymbol{x}_i \in R^n, t_i \in R, 1 \le i \le N\}$, activation functions $g(x)$, maximum hidden node number $L_{\max}$ and expected learning accuracy E_0.

Step1 Let the number of initial hidden nodes $k = 0$ and residual error $\boldsymbol{E} = \boldsymbol{t}$, where $\boldsymbol{t} = (t_1, t_2,\ldots,t_N)^{\mathrm{T}}$.
Step2 While $k \le L_{\max}$ and $\|\boldsymbol{E}\| \ge E_0$

1) Increase the number of hidden nodes k: $k = k + 1$;
2) Randomly generate one hidden node and calculate its output vector $\boldsymbol{h}_k$;
3) Calculate the output weight for the new hidden node

$$\beta_k = \frac{\langle \boldsymbol{E}, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{h}_k, \boldsymbol{h}_k \rangle}; \tag{5}$$

4) Calculate the new residual error $\boldsymbol{E} = \boldsymbol{E} - \beta_k \boldsymbol{h}_k$;
endwhile

It has been proved that SLFNs can work as universal approximators using I-ELM. However, I-ELM maybe can not let SLFNs learn the N distinct samples with any minor residual error within finite steps, because the output weight vector obtained by I-ELM is not the least squares solution of $\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}$. This will cause some issues such as generating redundant hidden nodes and reducing the convergence rate etc. In the next section, OI-ELM will be proposed to settle these issues thoroughly.

3 Proposed OI-ELM

In this section, the details of the proposed OI-ELM algorithm are described in both theory and application. Firstly, the principal ideas behind the OI-ELM algorithm are described in the next.

3.1 Principal ideas behind OI-ELM

Lemma 1 [21, Gram–Schmidt Algorithm] *Let $\boldsymbol{h}_1, \boldsymbol{h}_2,\ldots, \boldsymbol{h}_n$ be a linearly independent vector set in the inner product space V. Define vectors, $\boldsymbol{v}_1, \boldsymbol{v}_2,\ldots, \boldsymbol{v}_n$, recursively by the formula*

$$\boldsymbol{v}_k = \boldsymbol{h}_k - \frac{\langle \boldsymbol{v}_1, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{v}_1, \boldsymbol{v}_1 \rangle}\boldsymbol{v}_1 - \frac{\langle \boldsymbol{v}_2, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{v}_2, \boldsymbol{v}_2 \rangle}\boldsymbol{v}_2 - \cdots - \frac{\langle \boldsymbol{v}_{k-1}, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{v}_{k-1}, \boldsymbol{v}_{k-1} \rangle}\boldsymbol{v}_{k-1} \tag{6}$$

Then

1. *The vectors $\boldsymbol{v}_1, \boldsymbol{v}_2,\ldots, \boldsymbol{v}_n$ form an orthogonal set.*
2. *For each index $k = 1,2,\ldots,n$, $\mathrm{span}(\boldsymbol{h}_1, \boldsymbol{h}_2,\ldots, \boldsymbol{h}_k) = \mathrm{span}(\boldsymbol{v}_1, \boldsymbol{v}_2,\ldots, \boldsymbol{v}_k)$.*

Lemma 2: *Let $\boldsymbol{v}_1, \boldsymbol{v}_2,\ldots, \boldsymbol{v}_{k-1}$ form an orthogonal vector set in the inner product space V, if vector $\boldsymbol{h}_k$ can be expressed as a linear combination of $\boldsymbol{v}_1, \boldsymbol{v}_2,\ldots, \boldsymbol{v}_{k-1}$, then we have*

$$\boldsymbol{v}_k = \boldsymbol{h}_k - \frac{\langle \boldsymbol{v}_1, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{v}_1, \boldsymbol{v}_1 \rangle}\boldsymbol{v}_1 - \frac{\langle \boldsymbol{v}_2, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{v}_2, \boldsymbol{v}_2 \rangle}\boldsymbol{v}_2 - \cdots - \frac{\langle \boldsymbol{v}_{k-1}, \boldsymbol{h}_k \rangle}{\langle \boldsymbol{v}_{k-1}, \boldsymbol{v}_{k-1} \rangle}\boldsymbol{v}_{k-1} = 0 \tag{7}$$

Proof: Suppose there exists scalars $c_1, c_2,\ldots,c_{k-1}$ such that

$$\boldsymbol{h}_k = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_{k-1}\boldsymbol{v}_{k-1} \tag{8}$$

Substituting (8) into (7), we have $\boldsymbol{v}_k = 0$. □

Lemma 3 [6, Theorem 2.1]: *Given a standard SLFN with N hidden nodes and activation function g: R R which is infinitely differentiable in any interval, for N arbitrary distinct samples $(\boldsymbol{x}_i, \boldsymbol{t}_i)$, where $\boldsymbol{x}_i \forall R^n$, $\boldsymbol{t}_i \in R^m$, for any $\boldsymbol{w}_i$, b_i randomly chosen from any intervals of R^n and R, respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix $\boldsymbol{H}$ of the SLFN is invertible and $\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{T}\| = 0$.*

Remark 1: Lemma 3 implies that, with probability one, the column vectors of $\boldsymbol{H}$ can be made orthogonal by Gram–Schmidt orthogonalization method.

Therefore, based on lemma 1–lemma 3, the Gram–Schmidt orthogonalization method is incorporated into I-ELM to propose more effective OI-ELM as follows.

Algorithm OI-ELM Given N arbitrary distinct samples $\{(\boldsymbol{x}_i,\ \boldsymbol{t}_i)\ |\ \boldsymbol{x}_i \in R^n,\ \boldsymbol{t}_i \in R^m,\ 1 \le i \le N\}$, where $\boldsymbol{x}_i = (x_{i1}, x_{i2},\ldots,x_{in})^{\mathrm{T}}$, $\boldsymbol{t}_i = (t_{i1}, t_{i2},\ldots,t_{im})^{\mathrm{T}}$, $\boldsymbol{t} = (\boldsymbol{t}_1, \boldsymbol{t}_2,\ldots,\boldsymbol{t}_N)^{\mathrm{T}}$ activation functions $g(x)$, a small positive value ε, maximum number of iterations $L_{\max}$ and expected learning accuracy E_0.

Step1 Let the number of initial hidden nodes $k = 0$, the number of iterations $L = 0$ and residual error $\boldsymbol{E} = \boldsymbol{t}$.

(a) increase by one the number of hidden nodes and the number of iterations respectively k: $k = k + 1$, $L = L + 1$;

(b) Randomly generate one hidden node and calculate its output vector $\boldsymbol{h}_k$;

if $k = 1$

$\boldsymbol{v}_k = \boldsymbol{h}_k$;

else

$\boldsymbol{v}_k = \boldsymbol{h}_k - \frac{\langle \boldsymbol{v}_1, \boldsymbol{h}_k\rangle}{\langle \boldsymbol{v}_1, \boldsymbol{v}_1\rangle}\boldsymbol{v}_1 - \frac{\langle \boldsymbol{v}_2, \boldsymbol{h}_k\rangle}{\langle \boldsymbol{v}_2, \boldsymbol{v}_2\rangle}\boldsymbol{v}_2 - \cdots - \frac{\langle \boldsymbol{v}_{k-1}, \boldsymbol{h}_k\rangle}{\langle \boldsymbol{v}_{k-1}, \boldsymbol{v}_{k-1}\rangle}\boldsymbol{v}_{k-1}$

if $\|\boldsymbol{v}_k\| \ge \varepsilon$

Calculate the output weight for the new hidden node

$\boldsymbol{\beta}_k = \frac{\boldsymbol{v}_k^T \boldsymbol{E}}{\boldsymbol{v}_k^T \boldsymbol{v}_k}$

Calculate the new residual error $\boldsymbol{E} = \boldsymbol{E} - \boldsymbol{v}_k \boldsymbol{\beta}_k$

else

$k = k - 1$;

endwhile

Step2 while $L \le L_{\max}$ and $\|\boldsymbol{E}\| \ge E_0$

Remark 2: According to the Step 2 of OI-ELM and lemma 2, if the output vector of the new hidden node is almost linearly dependent with the output vectors of the existing hidden nodes, it will be given up. This means that OI-ELM can avoid the redundant hidden nodes and construct compact SLFNs in theory.

In the next subsection, we will rigorously prove that OI-ELM can obtain the least squares solution of $\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}$.

3.2 Analyses of the solution obtained by OI-ELM

Lemma 4: *Given N arbitrary distinct samples $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\ |\ \boldsymbol{x}_i \in R^n, \boldsymbol{t}_i \in R^m, 1 \le i \le N\}$ and a standard SLFN with L hidden nodes, $\boldsymbol{H}_L = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L] \in R^{n\times L}$ is the output matrix of the hidden layer, $\boldsymbol{\beta}_L = [\beta_1, \beta_2, \ldots \beta_L]^{\mathrm{T}} \in R^{L\times 1}$ is the weight vector connecting the hidden layer to the output layer. Let the error matrix $\boldsymbol{e}_L = \boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}_L$, $\boldsymbol{e}_0 = \boldsymbol{T}$, then $\boldsymbol{\beta}_L = \{\boldsymbol{\beta}^*|\min\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|,\ \boldsymbol{\beta}^* \in R^{L\times 1}\}$ holds only if*

$$\langle \boldsymbol{h}_i, \boldsymbol{h}_j\rangle = 0 \quad \text{for all } i \ne j \tag{9}$$

$$\text{and } \beta_i = \frac{\langle \boldsymbol{e}_{i-1}, \boldsymbol{h}_i\rangle}{\langle \boldsymbol{h}_i, \boldsymbol{h}_i\rangle} \quad \text{for all } 1 \le i \le L \tag{10}$$

Proof: We first prove that $\boldsymbol{e}_L \perp \text{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L\}$, and then, we further prove $\boldsymbol{\beta}_L = \{\boldsymbol{\beta}^*|\min\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|\}$.

(a) We can prove $\boldsymbol{e}_L \perp \text{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L\}$ by mathematical induction.

(a.1)

$$\begin{aligned}\langle \boldsymbol{e}_1, \boldsymbol{h}_1\rangle &= \langle \boldsymbol{T} - \beta_1\boldsymbol{h}_1, \boldsymbol{h}_1\rangle \\ &= \langle \boldsymbol{T}, \boldsymbol{h}_1\rangle - \beta_1\langle \boldsymbol{h}_1, \boldsymbol{h}_1\rangle\end{aligned} \tag{11}$$

Substituting (10) into (11), we have

$$\langle \boldsymbol{e}_1, \boldsymbol{h}_1\rangle = 0, \tag{12}$$

and

$$\begin{aligned}\langle \boldsymbol{e}_2, \boldsymbol{h}_1\rangle &= \langle \boldsymbol{e}_1 - \beta_2\boldsymbol{h}_2, \boldsymbol{h}_1\rangle \\ &= \langle \boldsymbol{e}_1, \boldsymbol{h}_1\rangle - \beta_2\langle \boldsymbol{h}_2, \boldsymbol{h}_1\rangle\end{aligned} \tag{13}$$

Substituting (9) and (12) into (13), we have

$$\langle \boldsymbol{e}_2, \boldsymbol{h}_1\rangle = 0, \tag{14}$$

and

$$\begin{aligned}\langle \boldsymbol{e}_2, \boldsymbol{h}_2\rangle &= \langle \boldsymbol{e}_1 - \beta_2\boldsymbol{h}_2, \boldsymbol{h}_2\rangle \\ &= \langle \boldsymbol{e}_1, \boldsymbol{h}_2\rangle - \beta_2\langle \boldsymbol{h}_2, \boldsymbol{h}_2\rangle\end{aligned} \tag{15}$$

Substituting (10) into (15), we have

$$\langle \boldsymbol{e}_2, \boldsymbol{h}_2\rangle = 0 \tag{16}$$

According to (14) and (16), it means that $\boldsymbol{e}_2 \perp \text{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2\}$.

(a.2)

Suppose for all $k \le L$,

$$\boldsymbol{e}_{k-1} \perp \text{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{k-1}\}, \tag{17}$$

$$\begin{aligned}\text{then} \langle \boldsymbol{e}_k, \boldsymbol{h}_k\rangle &= \langle \boldsymbol{e}_{k-1} - \beta_k\boldsymbol{h}_k, \boldsymbol{h}_k\rangle \\ &= \langle \boldsymbol{e}_{k-1}, \boldsymbol{h}_k\rangle - \beta_k\langle \boldsymbol{h}_k, \boldsymbol{h}_k\rangle = 0\end{aligned} \tag{18}$$

For all $j \le k - 1$

$$\begin{aligned}\langle \boldsymbol{e}_k, \boldsymbol{h}_j\rangle &= \langle \boldsymbol{e}_{k-1} - \beta_k\boldsymbol{h}_k, \boldsymbol{h}_j\rangle \\ &= \langle \boldsymbol{e}_{k-1}, \boldsymbol{h}_j\rangle - \beta_k\langle \boldsymbol{h}_k, \boldsymbol{h}_j\rangle\end{aligned} \tag{19}$$

Substituting (9) and (17) into (19), for all $j \le k - 1$, we have $\langle e_k, h_j\rangle = 0$

Thus $\boldsymbol{e}_k \perp \text{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2 \ldots \boldsymbol{h}_k\}$, that is, $\boldsymbol{e}_L \perp \text{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L\}$. Therefore

$$\boldsymbol{H}_L^T(\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}_L) = 0 \tag{20}$$

(b) According to (20), we have $(\boldsymbol{\beta}_L - \boldsymbol{\beta}^*)^{\mathrm{T}}\boldsymbol{H}_L^{\mathrm{T}}(\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}_L) = 0$, where $\boldsymbol{\beta}^* \in R^{L\times 1}$ is arbitrary. Let us consider $\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|$,

$$\begin{aligned}\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|^2 &= \|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}_L + \boldsymbol{H}_L(\boldsymbol{\beta}_L - \boldsymbol{\beta}^*)\|^2 \\ &= \|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}_L\|^2 + \|\boldsymbol{H}_L(\boldsymbol{\beta}_L - \boldsymbol{\beta}^*)\|^2 \\ &\geq \|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}_L\|^2\end{aligned} \tag{21}$$

$\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\| = \|\boldsymbol{H}_L\boldsymbol{\beta}_L - \boldsymbol{T}\|$ holds if only $\boldsymbol{\beta}_L = \boldsymbol{\beta}^*$. Therefore, $\boldsymbol{\beta}_L = \{\boldsymbol{\beta}^* | \min\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|, \boldsymbol{\beta}^* \in R^{L\times m}\}$. □

Furthermore, when $\boldsymbol{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots \boldsymbol{t}_m] \in R^{n\times m}$, we have $\boldsymbol{e}_L = [\boldsymbol{e}_{L1}, \boldsymbol{e}_{L2}, \ldots \boldsymbol{e}_{Lm}] \in R^{n\times m}$ and $\boldsymbol{e}_{Li} = \boldsymbol{t}_i - \boldsymbol{H}_L\boldsymbol{\beta}_{Li}$ for all $i \leq m$, where $\boldsymbol{\beta}_{Li}$ is the weight vector connecting the hidden layer to the ith output node. According to the proofs of Lemma 3, $\boldsymbol{e}_{Li} \in \mathrm{span}\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L\}$ is hold for all $i \leq m$, and therefore, Eqs. (20) and (21) are all hold.

Theorem 1: *Given a standard SLFN with L hidden nodes and activation function g(x): R R which is infinitely differentiable in any interval, for N arbitrary distinct samples* $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in R^n, \boldsymbol{t}_i \in R^m, 1 \leq i \leq N\}$ *and any* $\boldsymbol{w}_i$ *and* b_i *randomly generated from any intervals of* R^n *and R, respectively, define the output vector set of hidden nodes as* $\{\boldsymbol{h}_i = [g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_1 + b_i), g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_2 + b_i), \ldots, g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_N + b_i)]^{\mathrm{T}}, 1 \leq i \leq L\}$, *then there exists orthogonal vector set* $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L$ *such that span* $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L) =$ *span* $(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L)$ *and* $\boldsymbol{\beta} = \{\boldsymbol{\beta}^* | \min\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|, \boldsymbol{\beta}^* \in R^{L\times m}\}$ *with probability one, where* $\boldsymbol{H}_L = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L]$, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_L]^{\mathrm{T}}$, $\boldsymbol{\beta}_i = (\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{e}_{i-1})/(\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_i)$, $\boldsymbol{e}_i = \boldsymbol{T} - \boldsymbol{H}_i\boldsymbol{\beta}_i$ *for all* $1 \leq i \leq L$ *and* $\boldsymbol{e}_0 = \boldsymbol{T}$.

Proof: According to lemma 3, $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L$ can be made linearly independent with probability one.

According to lemma 1, there exists orthogonal vector set $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L$ such that $\mathrm{span}(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_L) = \mathrm{span}(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L)$.

According to lemma 4, $\boldsymbol{\beta} = \{\boldsymbol{\beta}^* | \min\|\boldsymbol{T} - \boldsymbol{H}_L\boldsymbol{\beta}^*\|, \boldsymbol{\beta}^* \in R^{L\times m}\}$ holds, which means that $\boldsymbol{\beta}$ is the least squares solution of $\boldsymbol{H}_L\boldsymbol{\beta}^* = \boldsymbol{T}$. This completes the proof. □

According to theorem 1, we can conclude that OI-ELM can obtain the least squares solution of $\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}$.

Moreover, the convergence of the proposed algorithm OI-ELM can be guaranteed by the following theorem 2.

3.3 Analyses of convergence

Lemma 5: *Suppose* $\boldsymbol{\beta}^*$ *is the least squares solution of* $\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T}$, $\boldsymbol{H}\boldsymbol{\beta}^* = \boldsymbol{T}$ *holds if only* $\boldsymbol{H}$ *is invertible.*

Proof: $\mathbf{H}\beta^* = \mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{T} = \mathbf{T}$. □

Theorem 2: (Convergence Theorem) *Given N arbitrary distinct samples* $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in R^n, \boldsymbol{t}_i \in R^m, 1 \leq i \leq N\}$ *and activation function g(x): R R which is infinitely differentiable in any interval, for any* $\boldsymbol{w}_i$ *and* b_i *randomly generated from any intervals of* R^n *and R, respectively, define the output vector set of hidden nodes as* $\{\boldsymbol{h}_i = [g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_1 + b_i), g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_2 + b_i), \ldots, g(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x}_N + b_i)]^{\mathrm{T}}, 1 \leq i \leq L\}$, *then for an arbitrary small positive value* γ, *there exists a positive* $L \leq N$ *such that* $\|\boldsymbol{e}_L\| < \gamma$ *with probability one, where* $\boldsymbol{e}_L = \boldsymbol{T} - \boldsymbol{h}_1\boldsymbol{\beta}_1 - \ldots - \boldsymbol{h}_L\boldsymbol{\beta}_L$, $\boldsymbol{e}_0 = \boldsymbol{T}$, $\boldsymbol{\beta}_i = (\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1})/(\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{h}_i)$.

Proof: Let $\Delta_i = \|\boldsymbol{e}_{i-1}\|^2 - \|\boldsymbol{e}_i\|^2$ for all $i \leq L$, then we have

$$\begin{aligned}\Delta_i &= \|\boldsymbol{e}_{i-1}\|^2 - \|\boldsymbol{e}_{i-1} - \boldsymbol{h}_i\boldsymbol{\beta}_i\|^2 \\ &= tr(\boldsymbol{e}_{i-1}^{\mathrm{T}}\boldsymbol{e}_{i-1}) - tr(\boldsymbol{e}_{i-1} - \boldsymbol{h}_i\boldsymbol{\beta}_i)^{\mathrm{T}}(\boldsymbol{e}_{i-1} - \boldsymbol{h}_i\boldsymbol{\beta}_i) \\ &= 2tr(\boldsymbol{e}_{i-1}^{\mathrm{T}}(\boldsymbol{h}_i\boldsymbol{\beta}_i)) - tr(\boldsymbol{h}_i\boldsymbol{\beta}_i)^{\mathrm{T}}(\boldsymbol{h}_i\boldsymbol{\beta}_i) \\ &= 2tr(\boldsymbol{e}_{i-1}^{\mathrm{T}}(\boldsymbol{h}_i\boldsymbol{\beta}_i)) - \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{h}_i tr(\boldsymbol{\beta}_i^{\mathrm{T}}\boldsymbol{\beta}_i) \\ &= \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{h}_i\left(\frac{tr((\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1})^{\mathrm{T}}(\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1}))}{(\boldsymbol{h}_i^{T}\boldsymbol{h}_i)^2}\right. \\ &\quad \left. - tr\left(\left(\beta_i - \frac{\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1}}{\boldsymbol{h}_i^{T}\boldsymbol{h}_i}\right)^{\mathrm{T}}\left(\boldsymbol{\beta}_i - \frac{\boldsymbol{h}_i^{T}\boldsymbol{e}_{i-1}}{\boldsymbol{h}_i^{T}\boldsymbol{h}_i}\right)\right)\right) \\ &= \frac{tr((\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1})^{\mathrm{T}}(\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{e}_{i-1}))}{\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{h}_i} \geq 0\end{aligned} \tag{22}$$

It means that $\{\|\boldsymbol{e}_i\|\}$ monotonically decreases with i. According to Lemma 3, the hidden layer output matrix $\boldsymbol{H}_N = [h_1, h_2, \ldots, h_N]$ is invertible with probability one. We have $\|\boldsymbol{e}_N\| = 0$ by Lemma 4 and Lemma 5. Therefore, given an arbitrary positive value γ, with probability one, there exists a positive $L \leq N$ such that $\|\boldsymbol{e}_L\| < \gamma$. □

Remark 3: In each learning step, $\Delta_i = \|\boldsymbol{e}_{i-1}\|^2 - \|\boldsymbol{e}_i\|^2$ is the largest because $\boldsymbol{\beta}_i$ is the least squares solution of $\boldsymbol{H}_i\ \boldsymbol{\beta}_i = \boldsymbol{T}$. It means that OI-ELM has the faster convergence rate than both I-ELM and EI-ELM in theory. So, OI-ELM will obtain more compact neural network architecture because of the fewer learning steps.

Remark 4: According to Theorem 1 and 2, OI-ELM can be applied in both regression and multiclass classification problems.

Remark 5: Theorem 2 implies that the hidden nodes in OI-ELM are not more than training samples.

Remark 6: According to [12] and [18], any bounded nonconstant piecewise continuous function can be selected as the activation function of OI-ELM.

4 Performance evaluation

In this section, the performance of OI-ELM is compared with other similar learning algorithms on nonlinear

dynamic system identification, ten real-world regression problems and five real-world multiclass classification problems. The datum of real-world cases come from the UCI database [22]. A simple sigmoid function $g(x) = 1/(1 + \exp(-x))$ is used as activation function for all the algorithms. In real-world experiments, all the inputs have been normalized into the range $[-1, 1]$ and the outputs in regression problems have been normalized into [0, 1]. The input weights and biases of the hidden nodes are randomly chosen from the interval $[-1, 1]$ for all experiments. All the simulations are carried out in MATLAB 7.0 environment running in a Pentium 4 with CPU 3.4 GHZ and 4.0 GB RAM. For each real-world case, fifty trials have been conducted for all algorithms, and the average results (AVE) and standard deviations (DEV) are listed in the corresponding tables, respectively.

4.1 Real-world regression problems

In this section, the performance of the proposed OI-ELM is compared with I-ELM and EI-ELM on ten benchmark real-world regression problems. Table 1 shows the specifications of these data sets. The expected training RMSEs of all algorithms are listed in Tables 2, 3 and 4. The maximum number of iterations or hidden nodes $L_{\max}$ equals to 100 for all algorithms and the parameter ε equals to 1e−6 for OI-ELM in every case. For all cases, the training sets are randomly generated from their whole sets before each trial of simulation, and the remainders form the testing sets. The average number of hidden nodes and training time of OI-ELM, I-ELM, EI-ELM are, respectively, shown in Table 2. Table 3 shows the average performances including training and testing RMSE of all algorithms. The winners in every comparing item have been shown in boldface in Tables 2 and 3. Table 4 shows the standard deviations of training and testing RMSE of each algorithm. In Table 4, the similar results of different algorithms are underlined and the outstanding results are shown in boldface.

Table 1 Specification of real-world classification cases

Data sets	Training sample	Testing sample	Attributes
Abalone	2,000	2,177	8
Census house (8L)	10,000	12,784	8
Stock	450	500	9
Servo	80	87	4
Breast cancer	100	94	32
Triazines	100	86	58
CPU	100	109	6
California housing	8,000	12,639	8
Boston housing	250	256	13
Auto price	80	79	14

Table 2 Comparison of network complexity and training time of OI-ELM, I-ELM and EI-ELM

Data sets (stop RMSE)	OI-ELM		I-ELM		EI-ELM	
	Nodes	Time(s)	Nodes	Time(s)	Nodes	Time(s)
Abalone (0.1)	**2.58**	**0.0062**	40.34	0.8237	8.10	0.1788
Census house (8L) (0.1)	**4.12**	**0.0312**	56.86	5.2479	11.38	0.7176
Stock (0.1)	**7.98**	**0.0271**	97.66	0.8383	66.82	0.9188
Servo (0.1)	**15.52**	**0.0555**	100	0.3114	100	0.5947
Breast Cancer (0.1)	**7.90**	**0.0331**	66.06	0.2324	19.42	0.1744
Triazines (0.1)	**58.88**	**0.2590**	100	0.3828	100	0.9348
CPU (0.1)	**2.64**	**0.0075**	13.60	0.0318	4.48	0.0434
California housing (0.2)	**4.22**	**0.0328**	69.58	6.0856	19.32	0.7173
Boston housing (0.1)	**21.44**	**0.0449**	100	0.5672	99.62	1.1591
Auto Price (0.1)	**7.12**	**0.0053**	49.76	0.0374	17.54	0.0424

It can be seen from Table 2 that OI-ELM constructs much more compact neural networks and obtains much smaller training time than both I-ELM and EI-ELM in all cases. It further demonstrates that OI-ELM can obtain much faster learning speed than the other two algorithms. As observed from Table 3, OI-ELM generally obtains much smaller training and testing RMSE in all cases than I-ELM. In most cases except for Triazines and CPU, OI-ELM achieves smaller training and testing RMSE than EI-ELM. As seen from Table 4, OI-ELM and EI-ELM have better stability than I-ELM in most cases.

4.2 Real-world multiclass classification problems

In this section, we mainly test the classification performance of the proposed algorithm by comparing with ELM[1] and I-ELM with the same number of hidden nodes. The comparisons of the performances have been conducted on five real-world multiclass classification problems including image segmentation, satellite image, letter recognition, glass and vehicle whose specifications are listed in Table 5. The parameter ε equals to 1e−6 for OI-ELM in this section, and the numbers of hidden nodes in all cases are listed in Table 6. For each case, the training set is randomly

[1] ELM Source Codes: http://www.ntu.edu.sg/home/egbhuang/.

Table 3 Comparison of the average of training and testing RMSE of OI-ELM, I-ELM and EI-ELM

Data sets (stop RMSE)	OI-ELM		I-ELM		EI-ELM	
	Training	Testing	Training	Testing	Training	Testing
Abalone (0.1)	**0.0939**	**0.0936**	0.1001	0.1002	0.0962	0.0969
Census house (8L) (0.1)	**0.0966**	**0.0973**	0.0998	0.1018	0.0996	0.0998
Stock (0.1)	**0.0910**	**0.0963**	0.1193	0.1202	0.0998	0.1010
Servo (0.1)	**0.0955**	**0.1390**	0.1765	0.1807	0.1682	0.1709
Breast cancer (0.1)	**0.0914**	**0.1084**	0.1032	0.1253	0.0984	0.1098
Triazines (0.1)	**0.0976**	0.2424	0.2104	0.2717	0.1437	**0.2149**
CPU (0.1)	**0.0837**	0.1075	0.0943	0.1151	0.0891	**0.1026**
California housing (0.2)	**0.1818**	**0.1827**	0.2001	0.2003	0.1988	0.1986
Boston housing (0.1)	**0.0982**	**0.1184**	0.1390	0.1421	0.1110	0.1186
Auto price (0.1)	**0.0927**	**0.1065**	0.0999	0.1093	0.0988	0.1073

Table 4 Comparison of the standard deviation of the training and testing RMSE of OI-ELM, I-ELM and EI-ELM

Data sets (stop RMSE)	OI-ELM		I-ELM		EI-ELM	
	Training	Testing	Training	Testing	Training	Testing
Abalone (0.1)	0.0044	0.0046	0.0047	0.0059	0.0038	0.0042
Census house (8L) (0.1)	0.0041	0.0053	5.5930e−004	0.0027	**4.1943e−5**	0.0031
Stock (0.1)	0.0069	0.0068	0.0143	0.0150	**5.6034e−4**	0.0041
Servo (0.1)	**0.0042**	0.0172	0.0142	0.0135	0.0123	0.0144
Breast cancer (0.1)	0.0071	0.0154	0.0083	0.0173	0.0018	0.0154
Triazines (0.1)	**0.0023**	0.0254	0.0366	0.1218	0.0126	0.0249
CPU (0.1)	0.0112	0.0253	**0.0069**	0.0277	0.0101	0.0272
California housing (0.2)	0.0099	0.0116	0.0025	0.0036	0.0025	0.0036
Boston housing (0.1)	0.0017	0.0090	0.0122	0.0160	0.0072	0.0085
Auto price (0.1)	**9.7304e−4**	0.0183	0.0038	0.0193	0.0022	0.0219

Table 5 Specification of real-world classification cases

Data sets	Training sample	Testing sample	Attributes	Classes
Image segmentation	1,100	1,210	18	7
Satellite image	4,400	2,035	36	6
Letter recognition	10,000	10,000	16	26
Glass	110	104	9	7
Vehicle	420	426	18	4

generated from its whole set before each trial of simulation, and the remainder forms the testing set. Table 6 shows the comparison results including training rate and testing rate. For all cases, the winners in each comparing item are shown in boldface in Table 6. It can be seen from Table 6 that OI-ELM obtains much larger training rate and testing rate in most cases than both ELM and I-ELM when they have the same number of hidden nodes. Furthermore, OI-ELM obtains much smaller standard deviations of training rate and testing rate than I-ELM. It means that OI-ELM is relatively stable than I-ELM in multiclass classification problems.

4.3 Nonlinear dynamic system identification

The nonlinear dynamic system to be identified is described as (23). It has been used in [23–25] to evaluate the performance of neural network algorithms.

$$y(t+1) = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t) \tag{23}$$

$t \in [1, 400],\ y(0) = 0,\ y(1) = 0,\ u(t) = \sin\left(\frac{2\pi t}{25}\right).$

The SLFN used in this test case has three inputs $y(t-1)$, $y(t)$, $u(t)$ and one output $y(t+1)$. A set of 100 input data in the interval [1,100] is chosen as training data and another set of 100 input data in the interval [301,400] is chosen as the testing data. The expected training root mean square error (RMSE) for every algorithm equals to 0.001. The maximum number of iterations or hidden nodes $L_{\max}$ equals to 200 for all algorithms and ε equals to 0.0001 for OI-ELM in this case. The results of comparison are shown in Figs. 1, 2 and 3.

The average of the training RMSE obtained by I-ELM, EI-ELM, CI-ELM and OI-ELM is shown in Fig. 1. It can be seen from Fig. 1 that OI-ELM can obtain much faster

Table 6 Performance comparison in classification problems

Data sets	Approaches	No. of hidden nodes	Training rate (%)		Testing rate (%)	
			Ave	Dev	Ave	Dev
Image segmentation	OI-ELM	200	0.9742	**0.0018**	**0.9451**	**0.0049**
	ELM	200	**0.9798**	0.0020	0.9407	0.0050
	I-ELM	200	0.8412	0.0116	0.8363	0.0113
Satellite image	OI-ELM	200	**0.9005**	**0.0026**	**0.9382**	**0.0018**
	ELM	200	0.9000	0.0029	0.8778	0.0028
	I-ELM	200	0.7648	0.0047	0.8883	0.0021
L-recognition	OI-ELM	200	**0.8318**	**0.0023**	**0.8278**	**0.0026**
	ELM	200	0.8301	0.0038	0.8176	0.0034
	I-ELM	200	0.5064	0.0122	0.5033	0.0117
Glass	OI-ELM	50	**0.9873**	**0.0047**	**0.7600**	**0.0254**
	ELM	50	0.9018	0.0141	0.6394	0.0343
	I-ELM	50	0.7073	0.0326	0.6745	0.0526
Vehicle	OI-ELM	100	**0.9107**	**0.0075**	0.7736	0.0171
	ELM	100	0.9076	0.0133	**0.7840**	**0.0140**
	I-ELM	100	0.6126	0.0168	0.5833	0.0153

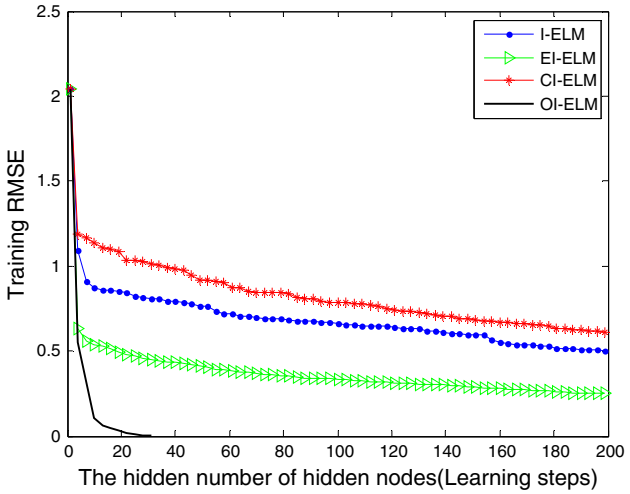

Fig. 1 Average training RMSE of different algorithms in nonlinear dynamic system identification case

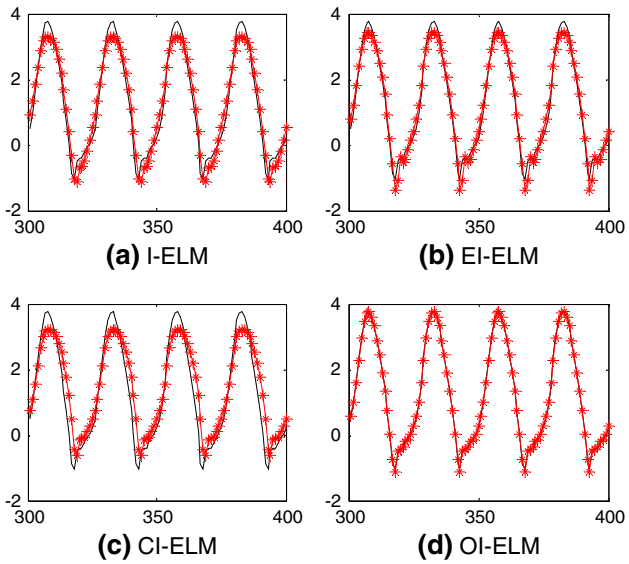

Fig. 2 Identifying result comparison of different algorithms in nonlinear dynamic system identification case (*straight line* represents the target; *dash line* represents the output of the SLFN)

learning rate and much fewer hidden nodes than the other three algorithms in the case of nonlinear dynamic system identification. Figure 2 shows the comparison of identifying results of different algorithms, from which OI-ELM has the best generalization performance on nonlinear dynamic system identification among the four methods. In order to test the stability of OI-ELM, one hundred trials have been conducted under the same conditions for OI-ELM. The number of hidden nodes and testing RMSE distributions are shown in Fig. 3. As observed from Fig. 3, the distribution of the number of hidden nodes and the testing RMSE of OI-ELM are very centralized, which means that OI-ELM is very stable. Figure 4 shows the growing process of hidden nodes along with the epoch, suggesting OI-ELM can avoid the redundant nodes in training process. In a word, it can be observed that the proposed OI-ELM can achieve faster training speeds, more compact network structures and better generalization performance than I-ELM, EI-ELM and CI-ELM in nonlinear dynamic system identification cases.

5 Conclusions

In this paper, we incorporate the Gram–Schmidt orthogonalization method into I-ELM to propose the orthogonal

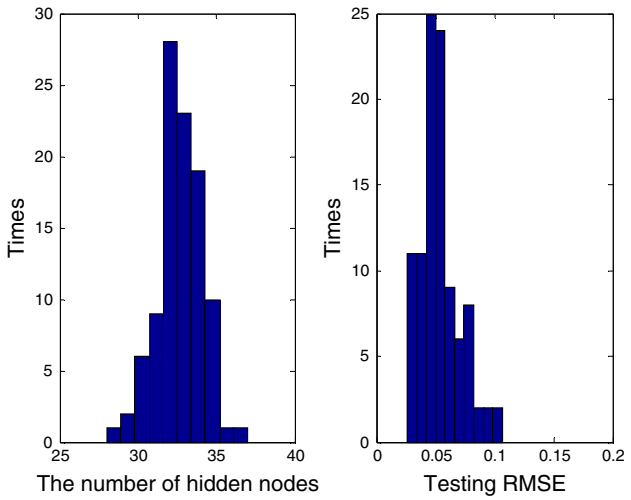

Fig. 3 The number of hidden nodes and testing RMSE distribution of OI-ELM in nonlinear dynamic system identification case

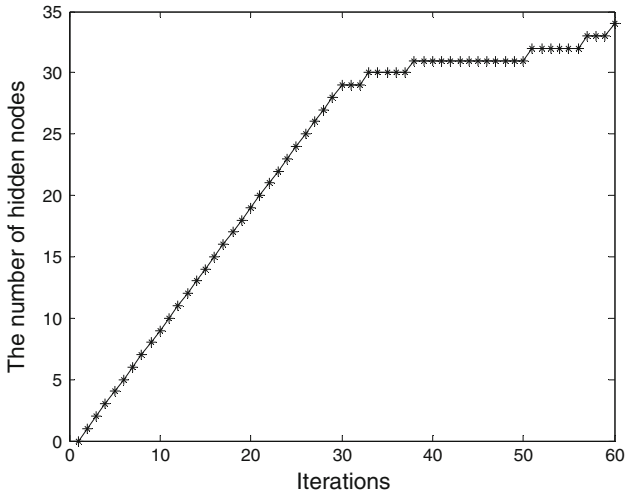

Fig. 4 The growing process of OI-ELM's hidden nodes along with the training epoch in nonlinear dynamic system identification case

incremental extreme learning machine (OI-ELM). Similar to I-ELM, OI-ELM also randomly generates hidden nodes one by one and analytically calculates the output weights by a simple formula. The main difference between I-ELM and OI-ELM is that OI-ELM can obtain the least squares solution of $\boldsymbol{H\beta} = \boldsymbol{T}$, which is rigorously proved in this paper. And the convergence of OI-ELM can be guaranteed in theory. Compared with ELM, I-ELM, CI-ELM and EI-ELM, OI-ELM can achieve much more compact network architectures at faster convergence rate, which are further verified by the simulation results on some benchmark problems including nonlinear dynamic system identification, ten real-world regression problems and five real-world multiclass classification problems.

References

1. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9(1):224–229
2. Hu X-f, Zhao Z, Wang S et al (2008) Multi-stage extreme learning machine for fault diagnosis on hydraulic tube tester. Neural Comput Appl 17(4):399–403
3. Kwok TY, Yeung DY (1997) Objective functions for training new hidden units in constructive neural networks. IEEE Trans Neural Netw 8(5):1131–1148
4. Teoh EJ, Tan KC, Xiang C (2006) Xiang. Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. IEEE Trans Neural Netw 17(6):1623–1629
5. Peng JX, Li K, Irwin GW (2008) A new jacobian matrix for optimal learning of single-layer neural networks. IEEE Trans Neural Netw 19(1):119–129
6. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
7. Liang NY, Huang GB, Saratchandran P et al (2006) A fast and accurate on-line sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17(6):1411–1423
8. Feng G, Huang GB, Lin Q et al (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
9. Miche Y, Sorjamaa A, Bas P et al (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
10. Soria-Olivas E, Gomez-Sanchis J, Martin JD et al (2011) BELM: Bayesian extreme learning machine. IEEE Trans Neural Netw 22(3):505–509
11. Zhang R, Lan Y, Huang GB et al (2012) Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Trans Neural Netw Learn Syst 23:365–371
12. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
13. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16):3056–3062
14. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16):3460–3468
15. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern 42(2):513–529
16. Tang X, Han M (2009) Partial Lanczos extreme learning machine for single-output regression problems. Neurocomputing 72(13):3066–3076
17. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cyber 2(2):107–122
18. Huang GB, Zhu QY, Mao KZ et al (2006) Can threshold networks be trained directly. IEEE Trans Circuits Syst II Express Briefs 53(3):187–191
19. Li G, Niu P (2013) An enhanced extreme learning machine based on ridge regression for regression. Neural Comput Appl 22(3–4):803–810
20. Meir R, Maiorov VE (2000) On the optimality of neural-network approximation using incremental algorithms. IEEE Trans Neural Netw 11(2):323–337
21. Shores TS (2007) Applied linear algebra and matrix analysis. Springer, Berlin
22. Bache K, Lichman M (2013) UCI machine learning repository. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml

23. Han HG, Qiao JF (2013) A structure optimisation algorithm for feedforward neural network construction. Neurocomputing 99:347–357
24. Han HG, Qiao JF (2010) Research on an online self-organizing radial basis function neural network. Neural Comput Appl 19(5):667–676
25. Leng G, McGinnity TM, Prasad G (2006) Design for self-organizing fuzzy neural networks based on genetic algorithms. IEEE Trans Fuzzy Syst 14(6):755–766