Wu YX, Liu D, Jiang H. Length-changeable incremental extreme learning machine. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(3): 630–643 May 2017. DOI 10.1007/s11390-017-1746-7

Length-Changeable Incremental Extreme Learning Machine

You-Xi Wu [1,2,3] , *Senior Member, CCF*, Dong Liu [1,3,4], and He Jiang [4,*], *Senior Member, CCF*

[1] *School of Computer Science and Software, Hebei University of Technology, Tianjin 300130, China*

[2] *School of Economics and Management, Hebei University of Technology, Tianjin 300130, China*

[3] *Hebei Province Key Laboratory of Big Data Calculation, Tianjin 300401, China*

[4] *School of Software, Dalian University of Technology, Dalian 116621, China*

E-mail: wuc@scse.hebut.edu.cn; dongliu05@gmail.com; jianghe@dlut.edu.cn

Received November 29, 2015; revised March 8, 2017.

Abstract Extreme learning machine (ELM) is a learning algorithm for generalized single-hidden-layer feed-forward networks (SLFNs). In order to obtain a suitable network architecture, Incremental Extreme Learning Machine (I-ELM) is a sort of ELM constructing SLFNs by adding hidden nodes one by one. Although kinds of I-ELM-class algorithms were proposed to improve the convergence rate or to obtain minimal training error, they do not change the construction way of I-ELM or face the over-fitting risk. Making the testing error converge quickly and stably therefore becomes an important issue. In this paper, we proposed a new incremental ELM which is referred to as Length-Changeable Incremental Extreme Learning Machine (LCI-ELM). It allows more than one hidden node to be added to the network and the existing network will be regarded as a whole in output weights tuning. The output weights of newly added hidden nodes are determined using a partial error-minimizing method. We prove that an SLFN constructed using LCI-ELM has approximation capability on a universal compact input set as well as on a finite training set. Experimental results demonstrate that LCI-ELM achieves higher convergence rate as well as lower over-fitting risk than some competitive I-ELM-class algorithms.

Keywords single-hidden-layer feed-forward network (SLFN), incremental extreme learning machine (I-ELM), random hidden node, convergence rate, universal approximation

1 Introduction

Among various kinds of neural networks, single-hidden-layer feed-forward networks (SLFNs) are extensively investigated in both theoretical and applied fields due to their simplicity and approximation capability. The output function of a single-output SLFN with d input nodes and L hidden nodes can be represented by

$$f(\boldsymbol{x}) = \sum_{j=1}^{L} \beta_j G(\boldsymbol{x}, \boldsymbol{a}_j, b_j),$$
$$\boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{a}_j \in \mathbb{R}^d, b_j \in \mathbb{R},$$

where β_j is the output weight of the connection between the j-th hidden node and the output node, $G(\boldsymbol{x}, \boldsymbol{a}_j, b_j)$ denotes the j-th hidden node output function, $\boldsymbol{x}$ is the input of the SLFN, $\boldsymbol{a}_j$ is the weight vector of the connection between the input layer and the j-th hidden node, and b_j is the bias of the j-th hidden node. The hidden nodes of SLFNs can be traditional additive hidden nodes and radial basis function (RBF) nodes[1-2]. Or they can be generalized to be non-neuron alike, such as support vector machines (SVMs)[3], ridge polynomial networks[4], polynomial fuzzy rules[5], and Fourier and wavelet networks[6-8]. Among them, two kinds of hidden nodes are usually mentioned.

1) Additive hidden nodes with activation function g:

$$G(\boldsymbol{x}, \boldsymbol{a}_j, b_j) = g(\boldsymbol{a}_j \cdot \boldsymbol{x} + b_j), \boldsymbol{a}_j \in \mathbb{R}^d, b_j \in \mathbb{R}.$$

2) RBF hidden nodes with activation function g:

$$G(\boldsymbol{x}, \boldsymbol{a}_j, b_j) = g(\|\boldsymbol{x} - \boldsymbol{a}_j\|_2 / b_j), \boldsymbol{a}_j \in \mathbb{R}^d, b_j \in \mathbb{R}^+.$$

Regular Paper

This work was partially supported by the National Natural Science Foundation of China under Grant Nos. 61673159 and 61370144, and the Natural Science Foundation of Hebei Province of China under Grant No. F2016202145.

*Corresponding Author

Theoretical studies show that any continuous function can be approximated by SLFN if all the parameters of SLFN are allowed to be tuned during learning[9-11]. However, the traditional gradient descent-based tuning methods are generally time consuming and easily converge to local minima. Different from the conventional neural network theories, Huang *et al.* proposed a new learning scheme for SLFNs: Extreme Learning Machine (ELM)[12-13]. In ELM, all the hidden node parameters ($\boldsymbol{a}_j$ and b_j) are randomly chosen, and only the weights connecting the hidden layer to the output node (β_j) need to be determined in training. What is more, the ELM learning algorithm could be used to train SLFNs with non-differentiable and even discontinuous activation functions. Its simplicity and efficiency has also been shown in many applications[14-20].

In order to obtain a suitable network architecture, many algorithms were proposed to adjust the hidden nodes of an ELM[21-30]. Some of them can be summarized as constructive algorithms, which start with a small network and then gradually add new hidden nodes, such as Incremental Extreme Learning Machine (I-ELM)[21], Constructive Multi-output Extreme Learning Machine (CM-ELM)[22], Error Minimized Extreme Learning Machine (EM-ELM)[23] and Constructive Parsimonious Extreme Learning Machine (CP-ELM)[24]. Some belong to destructive algorithms, which initiate a large network at beginning and then remove the redundant nodes, such as Optimally Pruned Extreme Learning Machine (OP-ELM)[25], Destructive Parsimonious Extreme Learning Machine (DP-ELM)[24] and Sparse Semi-Supervised Extreme Learning Machine (S3ELM)[26]. And some use an integration of the two methods[27-28].

Among them, I-ELM is a sort of ELM constructing SLFNs by adding hidden nodes one by one. In particular, Huang *et al.*[21] proved that SLFNs constructed by I-ELM can approximate any continuous target function under the meaning of L^2 on any compact input set. In I-ELM, the activation function of the hidden nodes can be any bounded non-constant piecewise continuous function (additive node) or any integrable piecewise continuous function (RBF node). Convex Incremental Extreme Learning Machine (CI-ELM)[31] and Enhanced random search based Incremental Extreme Learning Machine (EI-ELM)[32] are two improved algorithms based on the original I-ELM and they can achieve faster convergence rates and more compact network architectures. The approximation capability of CI-ELM and EI-ELM can be illustrated by the same theory as that of I-ELM with a slight modification. However, many learning steps are usually still needed for both CI-ELM and EI-ELM to converge. Feng *et al.* proposed another type of incremental ELM: EM-ELM[23]. In EM-ELM, one or more hidden nodes are allowed to be added into the network at a learning step and all the output weights are recalculated at each step. This means that the newly added hidden nodes exert a relatively large influence on the current network. Xu *et al.*[33] proposed Enhancement of Incremental Regularized Extreme Learning Machine (EIR-ELM) to improve the generalization performance of EM-ELM and EIR-ELM has the similar training time to EM-ELM. Different from them, we did not try to find any better hidden nodes and did not replace any added hidden nodes during learning in LCI-ELM. We focus on how to find a simple constructive way to make the testing errors converge quickly and stably.

In order to achieve further improvement in the convergence rate as well as keeping the testing error convergence stable, this paper proposes an improved form of I-ELM: Length-Changeable Incremental ELM (LCI-ELM). LCI-ELM allows several hidden nodes to be added to the network at a time and its requirements for hidden nodes are the same with those of I-ELM. The method of calculating the output weights is concise but does not meet the conditions for convergence in [31]. By modifying some procedures of the proof in [21], the universal approximation capability of LCI-ELM can be proved. Furthermore, the approximation of the training set can also be shown using a similar method.

This paper is organized as follows. Section 2 gives a description of the newly proposed LCI-ELM and looks at the ways to determine the output weights. Section 3 rigorously proves that LCI-ELM can work as a universal approximator for any continuous function under the meaning of L^2 in any compact set. Section 4 explains how to use LCI-ELM in practice and the LCI-ELM algorithm is given. This section also shows why the target output can be approximated using the proposed algorithm. A performance comparison of LCI-ELM, I-ELM, CI-ELM, EI-ELM ($k = 10, 20$), EM-ELM, and EIR-ELM is presented in Section 5. The discussions and conclusions are given in Section 6.

2 Proposed LCI-ELM

Without loss of generality, we assume that the network has only one output node. The analysis in this

paper can easily be extended to multiple output node cases. We choose f_n as the network output function after n ($n \geqslant 0$) learning steps and $g_j = G(\boldsymbol{x}, \boldsymbol{a}_j, b_j)$ as the output function of the j-th ($j \geqslant 1$) hidden node. We can simply initialize f_n with $f_0 = 0$. Let $e_n = f - f_n$ denote the residual error function for the current network output function f_n where f is the target function. Similar to [21, 31-32], all the functions mentioned above are discussed in space $L^2(X)$. The definition of $L^2(X)$ is given as follows.

Definition 1. *Space* $L^2(X) = \{f | f : X \to \mathbb{R}, \int_X |f(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} < \infty\}$, *where* X *is a compact subset of* $\mathbb{R}^d$. *For* $f, g \in L^2(X)$, *the inner product* $\langle f, g\rangle$ *is defined by* $\langle f, g\rangle = \int_X f(\boldsymbol{x}) g(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$, *and the norm of* f *in* $L^2(X)$ *space is denoted as* $\|f\| = (\int_X |f(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x})^{1/2}$.

In I-ELM, only one hidden node is added into the network in a learning step. The method of constructing SLFNs is

$$\begin{aligned} &f_{n+1} = f_n + \beta_{n+1} g_{n+1}, \\ &\beta_{n+1} = \langle f - f_n, g_{n+1}\rangle / \|g_{n+1}\|^2, \end{aligned} \tag{1}$$

where β_{n+1} is just the output weight of the newly added hidden node.

In this way, the output weights of the existing nodes remain unchanged when a new node is added. This means that adding a new node has minor impact on the current network output especially when the error e_n is small. The advantage of this method is that the network will exhibit reasonable performance when improper nodes are added. This point can also be verified by experiments[21]: the testing error usually has a decreasing trend while adding the hidden nodes. But the disadvantage is that it usually needs too many hidden nodes to reach a certain training accuracy. Some improvement algorithms of I-ELM, such as CI-ELM[31] and EI-ELM[32], improve the convergence rate. However, these algorithms do not give obviously better performance due to the similar way in which the I-ELM network is constructed.

Compared with this method, another ELM incremental constructive model called EM-ELM[23] uses an error minimization based method: once the hidden node output functions are given, the output weights of the hidden nodes are determined to make the norm of the training error vector reach the minimum. This means that all the former node output weights should be recalculated after adding a new hidden node to the network. EM-ELM has a higher convergence rate than I-ELM, and thus it usually needs fewer hidden nodes to reach a certain training accuracy. But a smaller training error does not always mean better generalization performance. Along with the increase in the number of hidden nodes, the training error decreases. But the testing error may increase or oscillate, thereby overfitting may happen sometimes[33].

In order to improve the convergence rate as well as keeping the network output stable during training, we consider a compromise of the above two models. We allow adding more than one hidden node to the network at a time. Assuming that the number of newly added nodes at step i is t_i, then the total number of hidden nodes after n steps is $S_n = \sum_{i=1}^{n} t_i$. In order to make the network output stable, the sequence $\{t_i\}$ should decrease and the lower bound of t_i is 1. In our experiments, we simply set the sequence to be a negative exponential form.

Then the hidden nodes of the existing network will be regarded as a whole in subsequent learning, that is, the relative ratio of the output weights remains unchanged while the output weights from then on are recalculated. The output function updates as follows.

$$f_{n+1} = w_0^{(n+1)} f_n + \sum_{j=1}^{t_{n+1}} w_j^{(n+1)} g_{S_n+j} \quad (n \geqslant 0), \tag{2}$$

where $\boldsymbol{w}^{(n+1)} = (w_0^{(n+1)}, w_1^{(n+1)}, \ldots, w_{t_{n+1}}^{(n+1)})^{\mathrm{T}}$ is chosen to minimize the new error $\|e_{n+1}\|$, equivalently, to minimize the function

$$\begin{aligned} &J_{n+1}(\boldsymbol{x}) = \|f - \boldsymbol{x} \cdot (f_n, g_{S_n+1}, \ldots, g_{S_{n+1}})\|^2, \\ &\boldsymbol{x} \in \mathbb{R}^{1+t_{n+1}}. \end{aligned} \tag{3}$$

Obviously, function $J(\boldsymbol{x})$ has a minimum, but its minimum value point may not be unique. According to the proof in Section 3, the universal approximation capability of SLFN always holds regardless of whichever minimum value point we choose as $\boldsymbol{w}$ is. We can choose any of the minimum value points and it can also be written as

$$\begin{aligned} &(w_0^{(n+1)}, w_1^{(n+1)}, \ldots, w_{t_{n+1}}^{(n+1)})^{\mathrm{T}} \\ &= \arg \min_{\boldsymbol{x} \in \mathbb{R}^{1+t_{n+1}}} \|f - \boldsymbol{x} \cdot (f_n, g_{S_n+1}, \ldots, g_{S_{n+1}})\|. \end{aligned}$$

Comparing (2) with (1), I-ELM would be a special case of LCI-ELM by setting both the coefficient $w_0^{(n+1)}$ and the number of the newly added hidden nodes t_{n+1} to 1. Generally, to add t_{n+1} ($t_{n+1} > 1$) hidden nodes into the network, I-ELM calculates the output weight of each new node one by one. But the weights are

calculated at one time to minimize the total error in LCI-ELM. LCI-ELM usually achieves lower training error at this phase. Therefore, allowing the addition of several nodes at a time, as well as using the partial error minimization method, can help to improve the convergence rate. The network output can be stable as the newly added nodes have a relatively small impact on it and the number of newly added nodes decreases each time. The output function of every hidden node is randomly generated like other ELMs. This way of constructing SLFNs can be regarded as a new improvement algorithm of I-ELM based on length-changeable incremental method. In this way, the output weights of the network are updated as

$$\beta_j^{(n+1)} = \begin{cases} w_0^{(n+1)}\beta_j^{(n)}, \text{ if } 1 \leqslant j \leqslant S_n, \\ w_{j-S_n}^{(n+1)}, \qquad \text{if } S_n < j \leqslant S_{n+1}, \end{cases}$$

where $\boldsymbol{\beta}^{(n)} = (\beta_1^{(n)}, \beta_2^{(n)}, \ldots, \beta_{S_n}^{(n)})^{\mathrm{T}}$ is the output weights after n learning steps.

3 Universal Approximation Capability of LCI-ELM

For the sake of readability, we provide the following definitions.

Definition 2. *A function $g : \mathbb{R} \to \mathbb{R}$ is said to be piecewise continuous if it has only a finite number of discontinuities in any interval and its left and right limits are defined (not necessarily equal) at each discontinuity.*

Definition 3. *Function spaces based on function $g : \mathbb{R} \to \mathbb{R}$ are such that $G'(g) = \{g(\boldsymbol{a} \cdot \boldsymbol{x} + b)|(\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}\}$ and $G''(g) = \{g(\|\boldsymbol{x} - \boldsymbol{a}\|_2/b)|(\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}^+\}$.*

The output functions of the hidden layer nodes are from these two spaces. They are additive functions and RBFs, respectively.

Definition 4. *The random variables $(\boldsymbol{a}, b)$ are said to be positive continuous random variables if they are continuous random variables and their probability density function $p(\boldsymbol{x})$ is positive on $\mathbb{R}^d \times \mathbb{R}$ or $\mathbb{R}^d \times \mathbb{R}^+$ almost everywhere.*

Definition 5. *The function sequence $\{g_n = g(\boldsymbol{a}_n \cdot \boldsymbol{x} + b_n)\}$ or $\{g_n = g(\|\boldsymbol{x} - \boldsymbol{a}_n\|_2/b_n)\}$ is said to be a random function sequence if the corresponding parameters $(\boldsymbol{a}_n, b_n)$ are positive random variables on $\mathbb{R}^d \times \mathbb{R}$ or $\mathbb{R}^d \times \mathbb{R}^+$. We assume the variables $(\boldsymbol{a}_j, b_j)(j = 1, 2, \ldots)$ are mutually independent.*

From the proof of I-ELM[21], the sequence $\{\|e_n\|\}$ converges to 0 with probability of 1 when some conditions are fulfilled. One of the sufficient conditions is $e_n \perp (e_{n-1} - e_n)$[31], but it does not hold for LCI-ELM. However, the universal approximation capability of LCI-ELM can be proved by using the original proof with some steps modified.

Some necessary lemmas are used in our proof.

Lemma 1[34]. *For any $f \in L^2(X)$ and $g \in L^2(X)$, $f + g \in L^2(X)$ holds.*

Lemma 2[10]. *Assuming function g is continuous almost everywhere and $\boldsymbol{\mu}$ is a non-negative finite measure on $\mathbb{R}^d$ with compact support, then* span$G'(g)$ *is dense in $L^p(\boldsymbol{\mu})$ for $1 \leqslant p < \infty$ if and only if g is not a polynomial.*

We can simply choose Lebesgue measure as $\boldsymbol{\mu}$ with support X and $p = 2$. Then Lemma 2 is related to the circumstance we are concerned with.

Lemma 3[11,21]. *Let $K : \mathbb{R}^d \to \mathbb{R}$ be an integrable bounded function such that K is continuous almost everywhere and $\int_{\mathbb{R}^d} K(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \neq 0$. Then* span$\{K((\boldsymbol{x} - \boldsymbol{a})/b)|(\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}^+\}$ *is dense in $L^p(\mathbb{R}^d)$ for every $p \in [1, \infty)$.*

Lemma 4[21]. *Given a bounded non-constant piecewise continuous function $g : \mathbb{R} \to \mathbb{R}$, we have*①

$$\lim_{(\boldsymbol{a},b)\to(\boldsymbol{a}_0,b_0)} \|g(\boldsymbol{a} \cdot \boldsymbol{x} + b) - g(\boldsymbol{a}_0 \cdot \boldsymbol{x} + b_0)\| = 0,$$
$$(\boldsymbol{a}_0, b_0) \in \mathbb{R}^d \times \mathbb{R},$$

and

$$\lim_{(\boldsymbol{a},b)\to(\boldsymbol{a}_0,b_0)} \|g(\|\boldsymbol{x} - \boldsymbol{a}\|_2/b) - g(\|\boldsymbol{x} - \boldsymbol{a}_0\|_2/b_0)\| = 0,$$
$$(\boldsymbol{a}_0, b_0) \in \mathbb{R}^d \times \mathbb{R}^+.$$

Lemma 5. *Given a bounded non-constant piecewise continuous function $g : \mathbb{R} \to \mathbb{R}$, any given positive value $\hat{\theta} < \pi/2$ and any subsequence $\{n_k\}$ of nature number sequence, for any random function sequence $\{g_n\}$ and any $g_0 \in G'(g)\backslash\{g(b)|b$ is a discontinuity of $g\}$ or $g_0 \in G''(g)$, a $g_{n_i}(i \in \boldsymbol{N}_+)$ exists satisfying $\theta(g_{n_i}, g_0) < \hat{\theta}$ with probability of 1, where $\theta(g_{n_i}, g_0)$ is the angle formed by g_{n_i} and g_0.*

Proof. With the proof of Lemma 6 in [21], according to Lemma 4, $\delta > 0$ exists such that any $(\boldsymbol{a}, b) \in Q = \{(\boldsymbol{a}, b)|\|(\boldsymbol{a}, b) - (\boldsymbol{a}_0, b_0)\|_2 < \delta\}$, $\|g - g_0\| < \|g_0\| \sin(\hat{\theta})$.

Supposing that $p(\boldsymbol{x})$ is the probability density function of $(\boldsymbol{a}, b)$ where $\boldsymbol{x} \in \mathbb{R}^d \times \mathbb{R}$ for additive node case and $\boldsymbol{x} \in \mathbb{R}^d \times \mathbb{R}^+$ for RBF node case, as $\boldsymbol{a}$ and b

①The situation that $\boldsymbol{a}_0 = 0$ and $g(x)$ is discontinuous at $x = b_0$ should be excluded for additive functions. It is ignored in the original description[21].

are positive random variables, $p_0 = P\{(\boldsymbol{a}, b) \in Q\} = \int_Q p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} > 0$ holds.

The probability that an element in $\{g_{n_1}, g_{n_2}, \ldots, g_{n_M}\}$ belonging to Q exists is that $P_M = 1 - \prod_{i=1}^{M}(1-p_0) = 1-(1-p_0)^M$, then $P_\infty = 1$. Thus a $g_{n_i}(i \in \boldsymbol{N}_+)$ exists such that $g_{n_i} \in Q$ with probability of 1. Meanwhile, we can see that $2\langle g_{n_i}, g_0\rangle > \|g_{n_i}\|^2 + \|g_0\|^2\cos^2(\hat{\theta}) \geqslant 2\|g_{n_i}\|\|g_0\|\cos(\hat{\theta})$, that is, $\cos(\theta(g_{n_i}, g_0)) > \cos(\hat{\theta})$ where $0 \leqslant \theta(g_{n_i}, g_0) \leqslant \pi$, thereby $\theta(g_{n_i}, g_0) < \hat{\theta}$ holds with probability of 1. □

Then we can declare the universal approximation capability of LCI-ELM. It shows as follows.

Theorem 1. *Given any bounded non-constant piecewise continuous function $g : \mathbb{R} \to \mathbb{R}$ for additive nodes or any integrable piecewise continuous function $g : \mathbb{R} \to \mathbb{R}$ and $\int_{\mathbb{R}} g(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \neq 0$ for RBF nodes, for any continuous target function f and any random function sequence $\{g_n\}$, then $\lim_{n\to\infty}\|f - f_n\| = 0$ holds with probability of 1 if the function sequence $\{f_n\}$ generates as follows.*

1) $f_0 = 0$;

2) $f_{n+1} = w_0^{(n+1)} f_n + \sum_{j=1}^{t_{n+1}} w_j^{(n+1)} g_{S_{n+j}}$,

where $t_n \in \boldsymbol{N}_+$ (non-zero natural numbers), $S_n = \sum_{i=1}^{n} t_i$, $n \geqslant 1$ and $(w_0^{(n+1)}, w_1^{(n+1)}, \ldots, w_{t_{n+1}}^{(n+1)})^{\mathrm{T}} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^{1+t_{n+1}}}\|f - \boldsymbol{x}\cdot(f_n, g_{S_n+1}, \ldots, g_{S_{n+1}})\|$.

Proof. Since g is piecewise continuous, $f_0 = 0$ and f are continuous, we can easily have $f_0, f, g_0 \in L^2(X)$. And $\{f_n\}$ is constructed by addition, according to Lemma 1, $\forall n \in \boldsymbol{N}_+$, $f_n \in L^2(X)$ and $e_n \in L^2(X)$. We should note

$$\begin{aligned}
&\|e_n\| \\
&= \|f - f_n\| \\
&= \|f - (1, 0, \ldots, 0)\cdot(f_n, g_{S_{n+1}}, \ldots, g_{S_{n+1}})\| \\
&\geqslant \min_{\boldsymbol{x}\in\mathbb{R}^{1+t_{n+1}}}\|f - \boldsymbol{x}\cdot(f_n, g_{S_n+1}, \ldots, g_{S_{n+1}})\| \\
&= \|f - f_{n+1}\| \\
&= \|e_{n+1}\|,
\end{aligned}$$

which means that the error sequence $\{\|e_n\|\}$ is decreasing with lower bound being 0. Thus it has a limit which is not negative. Similar to [21], we will prove the limit is 0 by contradiction method.

Supposing that $\lim_{n\to\infty}\|e_n\| = r > 0$, we have $\forall n \in \boldsymbol{N}_+$, $\|e_0\| \geqslant \|e_n\| \geqslant r$. This implies that the sequence $\{e_n\}$ is covered by a compact set; thus, there exists a convergent sub-sequence $\{e_{n_k}\}$ such that $\lim_{k\to\infty} e_{n_k} = e^*$ and $\|e^*\| = r$. And $L^2(X)$ is complete, thereby $e^* \in L^2(X)$. Similar to the proof b.1) in [21], $g^* \in G'(g)$ or $g^* \in G''(g)$ exists such that g^* is not orthogonal to e^*. Otherwise, e^* is orthogonal to span$G'(g)$ or span$G''(g)$. As span$G'(g)$ or span$G''(g)$ is dense in $L^2(X)$ according to Lemma 2 and Lemma 3, there exists $e^{**} \in$ span$G'(g)$ or span$G''(g)$ that $\|e^{**} - e^*\| < r/2$. Thus $\langle e^*, e^{**}\rangle = \|e^*\|^2 + \langle e^*, e^{**} - e^*\rangle \geqslant \|e^*\|^2 - \|e^*\|\|e^{**} - e^*\| > r^2/2 > 0$, which is contradictory to the above conclusion that e^* is orthogonal to e^{**}, meaning that $0 \leqslant \sin(\theta(g^*, e^*)) < 1$. If $g^* = g(b^*)$ and g is discontinuous at $x = b^*$, we can simply choose $g^{**} = g(b^{**})$, where g is continuous at $x = b^{**}$, instead of g^*. This is because $\theta(g^*, e^*) = \theta(1, e^*) = \theta(g^{**}, e^*)$.

Then $\cos(\theta(e_{n_k}, e^*)) = (\|e_{n_k}\|^2 + \|e^*\|^2 - \|e_{n_k} - e^*\|^2)/2\|e_{n_k}\|\|e^*\| \to 1$, that is, $\theta(e_{n_k}, e^*) \to 0$ and we can choose a positive number $\xi = (1 - \sin(\theta(g^*, e^*)))/2 < 1$. According to the properties of limit, $\|e_{n_k}\| < r/(1-\xi)$ holds while $k > N_1$ and $\sin(\theta(e_{n_k}, e^*)) < \xi/2$ holds while $k > N_2$. Let $N = \max\{N_1, N_2\}$, and we have a sub-sequence of the nature number sequence by ranking the numbers in $\{i | S_{n_k} + 1 \leqslant i \leqslant S_{n_k+1}, k \geqslant N+1\}$ in order from small to large. We call it $\{n_k^*\}$. For any random function sequence $\{g_n\}$, $g_{n_i^*}(i \in \boldsymbol{N}_+)$ exists satisfying $\theta(g_{n_i^*}, g^*) < \arcsin(\xi/2)$ with probability of 1, according to Lemma 5, that is, $\sin(\theta(g_{n_i^*}, g^*)) < \xi/2$. We assume that $S_{n_p} + 1 \leqslant n_i^* \leqslant S_{n_p+1}$ where $p \geqslant N+1$, then

$$\begin{aligned}
&\|e_{n_p+1}\| \\
&= \min_{\boldsymbol{x}\in\mathbb{R}^{1+t_{n_p+1}}}\|f - \boldsymbol{x}\cdot(f_{n_p}, g_{S_{n_p}+1}, \ldots, g_{S_{n_p+1}})\| \\
&\leqslant \|f - f_{n_p} - \langle e_{n_p}, g_{n_i^*}\rangle g_{n_i^*}/\|g_{n_i^*}\|^2\| \\
&= \|e_{n_p}\|\sin(\theta(e_{n_p}, g_{n_i^*})) \\
&\leqslant \|e_{n_p}\|(\sin(\theta(e_{n_p}, e^*)) + \sin(\theta(e^*, g^*)) + \\
&\quad \sin(\theta(g^*, g_{n_i^*}))) \\
&< r(\xi/2 + 1 - 2\xi + \xi/2)/(1-\xi) \\
&= r,
\end{aligned}$$

which is contradictory to $\forall n \in \boldsymbol{N}_+$, $\|e_n\| \geqslant r$. Thereby $\lim_{n\to\infty}\|f - f_n\| = \lim_{n\to\infty}\|e_n\| = 0$. □

4 LCI-ELM in Real Applications

4.1 Algorithm Based on the Training Set

In LCI-ELM, the output weights are determined based on the target function f. However, the target function f is unavailable in practice and the training

samples cannot cover all the information. Similar to [21] and [35], we can use some vectors based on the training set instead of functions. We should make a transform from $L^2(X)$ to $\mathbb{R}^N$. For the training set $\{(\boldsymbol{x}_j, y_j)|1 \leqslant j \leqslant N\}$, let $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^{\mathrm{T}}$, $\boldsymbol{g}_n = (g_n(\boldsymbol{x}_1), g_n(\boldsymbol{x}_2), \ldots, g_n(\boldsymbol{x}_N))^{\mathrm{T}}$ and $\boldsymbol{f}_0 = (0, 0, \ldots, 0)^{\mathrm{T}}$. The vector norm $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}$ is used instead of the L^2 norm. The output weights are calculated as follows.

$$\boldsymbol{f}_{n+1} = w_0^{(n+1)}\boldsymbol{f}_n + \sum_{j=1}^{t_{n+1}} w_j^{(n+1)}\boldsymbol{g}_{S_n+j}, \tag{4}$$

$$\begin{aligned}\boldsymbol{w}^{(n+1)} &= (w_0^{(n+1)}, w_1^{(n+1)}, \ldots, w_{t_{n+1}}^{(n+1)})^{\mathrm{T}} \\ &= [\boldsymbol{f}_n, \boldsymbol{g}_{S_n+1}, \ldots, \boldsymbol{g}_{S_{n+1}}]^{+}\boldsymbol{y},\end{aligned} \tag{5}$$

where A^+ is the Moore-Penrose generalized inverse of A. According to the theory of generalized inverse, the solution presented by (5) uniquely exists and it makes (3) achieve its minimum.

LCI-ELM allows more than one hidden node to be added at a time and we set the number of newly added hidden nodes at step i as follows.

$$t_i = \lfloor ke^{-\lambda(i-1)} \rfloor + 1, i \in \boldsymbol{N}_+,$$

where $\lfloor \cdot \rfloor$ denotes the bottom integral function. The initial network has $k+1$ hidden nodes and the incremental number of hidden nodes decreases with lower bound being 1. Parameter λ shows the decreasing rate of the incremental number: the larger λ is, the higher the decreasing rate achieved by the incremental number, and the incremental number reduces to 1 while $i > \ln(k)/\lambda + 1$.

Obviously, the training error $\|\boldsymbol{e}_n\| = \|\boldsymbol{y} - \boldsymbol{f}_n\|$ decreases in the learning phase. In real applications, it is not usually easy or necessary to reach zero approximation error. Therefore the expected learning accuracy ϵ and the maximum hidden node number $L_{\max}$ are given at the same time. Thus, the LCI-ELM algorithm is summarized in Algorithm 1.

Algorithm 1. LCI-ELM

Require: a training set $\{(\boldsymbol{x}_j, y_j)|\boldsymbol{x}_j \in \mathbb{R}^d, y_j \in \mathbb{R}, 1 \leqslant j \leqslant N\}$, activation function $g(x)$, hidden node incremental parameters (k, λ), maximum number of hidden nodes $N_{\max}$, and expected learning accuracy ϵ

Ensure: the hidden nodes parameters $(\boldsymbol{a}_j, b_j)(1 \leqslant j \leqslant L)$ and the output weights $\boldsymbol{\beta}$

1: Initialization: $i = 1$, $L = 0$, target $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^{\mathrm{T}}$, $\boldsymbol{f}_0 = (0, 0, \ldots, 0)^{\mathrm{T}}$, residual error $\boldsymbol{e} = \boldsymbol{y}$ and output weights $\boldsymbol{\beta} = \phi$

2: **While** $L < L_{\max}$ **and** $\|\boldsymbol{e}\| > \epsilon$

3: Calculate the incremental number of new hidden nodes: $t = \lfloor ke^{-\lambda(i-1)} \rfloor + 1$

4: Randomly generate parameters $(\boldsymbol{a}_j, b_j)$ $(L+1 \leqslant j \leqslant L+t)$ for the new hidden nodes

5: Calculate the output functions of the new hidden nodes: $\boldsymbol{g}_j = (g_j(\boldsymbol{x}_1), g_j(\boldsymbol{x}_2), \ldots, g_j(\boldsymbol{x}_N))^{\mathrm{T}} (L+1 \leqslant j \leqslant L+t)$

6: Calculate the output weights of the new hidden nodes: $(w_0, w_1, \ldots, w_t)^{\mathrm{T}} = [\boldsymbol{f}, \boldsymbol{g}_{L+1}, \ldots, \boldsymbol{g}_{L+t}]^{+}\boldsymbol{y}$ and modify the output weights: $\boldsymbol{\beta} := (w_0\boldsymbol{\beta}^{\mathrm{T}}, w_1, w_2, \ldots, w_t)^{\mathrm{T}}$

7: Update the network output function $\boldsymbol{f} := w_0\boldsymbol{f} + \sum_{j=1}^{t} w_j\boldsymbol{g}_{L+j}$, residual error $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{f}$, $L := L + t$ and $i := i + 1$

8: **End While**

4.2 Approximation in the Training Set

We have shown the approximation capability of LCI-ELM from the universal approximation point of view. What is more, similar properties are maintained for the approximation aspect in the training set, that is, the target output $\boldsymbol{y}$ can be approximated by the output $\boldsymbol{f}$ in the LCI-ELM algorithm if the activation function $g(x)$ is infinitely differentiable and meets some other conditions.

Lemma 6. *Let $g : \mathbb{R} \to \mathbb{R}$ be an infinitely differentiable function on $\mathbb{R}$ and $g^{(k)}(x) \neq 0$ holds in some interval for any $k \geqslant 0$. Then space $S = \mathrm{span}\{(g(\boldsymbol{a}\cdot\boldsymbol{x}_1 + b), g(\boldsymbol{a}\cdot\boldsymbol{x}_2 + b), \ldots, g(\boldsymbol{a}\cdot\boldsymbol{x}_N + b))^{\mathrm{T}} | (\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}\}$ has dimension N for any distinct vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d$.* *Proof.* It is clear that space S is a subspace of $\mathbb{R}^N$, and thus its dimension is not larger than N.

Suppose the dimension is less than N, then a nonzero vector $\boldsymbol{c} \in \mathbb{R}^N$ exists and is orthogonal to S, that is, $\sum_{i=1}^{N} c_i g(\boldsymbol{a}\cdot\boldsymbol{x}_i + b) = 0$ for any $(\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}$. Let $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^{\mathrm{T}}$ and $\boldsymbol{a} = (a_1, a_2, \ldots, a_d)^{\mathrm{T}}$, then $\partial^{N-1} g(\boldsymbol{a}\cdot\boldsymbol{x}_i + b)/\partial {a_j}^k \partial b^{N-1-k} = {x_{ij}}^k g^{(N-1)}(\boldsymbol{a}\cdot\boldsymbol{x}_i + b)$ as g is infinitely differentiable. Therefore we have $\sum_{i=1}^{N} c_i {x_{ij}}^k g^{(N-1)}(\boldsymbol{a}\cdot\boldsymbol{x}_i + b) = 0$ for $1 \leqslant j \leqslant d$ and $0 \leqslant k \leqslant N-1$. Defining a set $I(n, z) = \{i|1 \leqslant i \leqslant N, x_{in} = z\}$, then the equation above can be written as $\sum_{t=1}^{r} {z_t}^k \sum_{i \in I(j, z_t)} c_i g^{(N-1)}(\boldsymbol{a}\cdot\boldsymbol{x}_i + b) = 0$ where $\{z_1, z_2, \ldots, z_r\}$ is made up of all the values of $x_{1j}, x_{2j}, \ldots, x_{Nj}$. As numbers $z_1, z_2, \ldots, z_r$ are different from each other, the Vandermonde determinant $\det[{z_j}^{i-1}]_{r\times r} \neq 0$. Thus $\sum_{i \in I(j, z_t)} c_i g^{(N-1)}(\boldsymbol{a}\cdot\boldsymbol{x}_i + b) = 0$ for $1 \leqslant t \leqslant r$. We can choose $j = 1$ and $z_t = x_{s1}$ and obviously $s \in I(1, x_{s1})$ holds for $1 \leqslant s \leqslant N$. Furthermore, we have $\sum_{i \in I(1, x_{s1})} c_i {x_{i2}}^k g^{(2(N-1))}(\boldsymbol{a}\cdot\boldsymbol{x}_i + b) = 0$,

and then $\sum_{i\in I(1,x_{s1})\cap I(2,x_{s2})} c_i g^{(2(N-1))}(\boldsymbol{a}\cdot\boldsymbol{x}_i+b)=0$.

Conducting this process d times, we have $\sum_{i\in\bigcap_{j=1}^{d} I(j,x_{sj})} c_i g^{(d(N-1))}(\boldsymbol{a}\cdot\boldsymbol{x}_i+b)=0$. For every element on the left of the equality, its subscript i satisfies $x_{ij}=x_{sj}(j=1,2,\ldots,d)$, that is, $\boldsymbol{x}_i=\boldsymbol{x}_s$. Therefore $i=s$ as $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N$ are distinct vectors. According to our assumption, $g^{(d(N-1))}(x)\neq 0$ in some interval. We can easily find $(\boldsymbol{a},b)\in\mathbb{R}^d\times\mathbb{R}$ such that $g^{(d(N-1))}(\boldsymbol{a}\cdot\boldsymbol{x}_s+b)\neq 0$. Thus $c_s=0$ holds for $s=1,2,\ldots,N$. This means that $\boldsymbol{c}=\boldsymbol{0}$ which is contradictory to $\boldsymbol{c}$ being non-zero. Thus $\dim S=N$. □

Note that the sigmoid function $y=1/(1+e^{-x})$ satisfies the conditions of Lemma 6.

Lemma 7[36]. *Suppose F' is completely monotonic (that is $(-1)^k F^{(k+1)}\geqslant 0$ on $(0,+\infty)$ for any $k\in\boldsymbol{N}_+$) but not constant on $(0,+\infty)$, F is continuous on $[0,+\infty)$ and positive on $(0,+\infty)$. Then for any distinct vectors $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N\in\mathbb{R}^d$ (d arbitrary)*

$$(-1)^{N-1}\det[F(\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2{}^2)]_{N\times N}>0.$$

Lemma 8. *Given a function $g:\mathbb{R}\to\mathbb{R}$, let $f(x)=g(\sqrt{x})$. Suppose f is continuous on $[0,+\infty)$ and positive on $(0,+\infty)$, f' is completely monotonic and not constant on $(0,+\infty)$. Then space $S=\mathrm{span}\{(g(\|\boldsymbol{x}_1-\boldsymbol{a}\|_2/b),g(\|\boldsymbol{x}_2-\boldsymbol{a}\|_2/b),\ldots,g(\|\boldsymbol{x}_N-\boldsymbol{a}\|_2/b))^{\mathrm{T}}|(\boldsymbol{a},b)\in\mathbb{R}^d\times\mathbb{R}^+\}$ has dimension N for any distinct vectors $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N\in\mathbb{R}^d$.*

Proof. We can also see that $\dim S\leqslant N$. According to Lemma 7, $(-1)^{N-1}\det[f(\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2{}^2)]_{N\times N}>0$, that is, $(-1)^{N-1}\det[g(\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2)]_{N\times N}>0$. Let $\boldsymbol{s}_i=(g(\|\boldsymbol{x}_1-\boldsymbol{x}_i\|_2),g(\|\boldsymbol{x}_2-\boldsymbol{x}_i\|_2),\ldots,g(\|\boldsymbol{x}_N-\boldsymbol{x}_i\|_2))^{\mathrm{T}}$, then $\det[\boldsymbol{s}_1,\boldsymbol{s}_2,\ldots,\boldsymbol{s}_N]\neq 0$. Thus vectors $\boldsymbol{s}_1,\boldsymbol{s}_2,\ldots,\boldsymbol{s}_N$ are linearly independent and $\boldsymbol{s}_i\in S$ (just to choose $\boldsymbol{a}=\boldsymbol{x}_i$ and $b=1$) for $i=1,2,\ldots,N$. Thus, $\dim S\geqslant N$. Finally we have $\dim S=N$. □

We can see that the multi-quadric function $y=(1+x^2)^{1/2}$ is qualified for Lemma 8.

Lemma 6 and Lemma 8 imply that $\mathrm{span}\{(G(\boldsymbol{x}_1,\boldsymbol{a},b),G(\boldsymbol{x}_2,\boldsymbol{a},b),\ldots,G(\boldsymbol{x}_N,\boldsymbol{a},b))^{\mathrm{T}}|(\boldsymbol{a},b)\in\mathbb{R}^d\times\mathbb{R}\}$ is dense in $\mathbb{R}^N$.

Lemma 9. *Given an infinitely differentiable function $g:\mathbb{R}\to\mathbb{R}$, let vector $\boldsymbol{g'}(\boldsymbol{a},b)$ denote $(g(\boldsymbol{a}\cdot\boldsymbol{x}_1+b),g(\boldsymbol{a}\cdot\boldsymbol{x}_2+b),\ldots,g(\boldsymbol{a}\cdot\boldsymbol{x}_N+b))^{\mathrm{T}}$ and $\boldsymbol{g''}(\boldsymbol{a},b)$ denote $(g(\|\boldsymbol{x}_1-\boldsymbol{a}\|_2/b),g(\|\boldsymbol{x}_2-\boldsymbol{a}\|_2/b),\ldots,g(\|\boldsymbol{x}_N-\boldsymbol{a}\|_2/b))^{\mathrm{T}}$, then we have*

$$\lim_{(\boldsymbol{a},b)\to(\boldsymbol{a}_0,b_0)}\|\boldsymbol{g'}(\boldsymbol{a},b)-\boldsymbol{g'}(\boldsymbol{a}_0,b_0)\|_2=0,$$

$$(\boldsymbol{a}_0,b_0)\in\mathbb{R}^d\times\mathbb{R},$$

and

$$\lim_{(\boldsymbol{a},b)\to(\boldsymbol{a}_0,b_0)}\|\boldsymbol{g''}(\boldsymbol{a},b)-\boldsymbol{g''}(\boldsymbol{a}_0,b_0)\|_2=0,$$

$$(\boldsymbol{a}_0,b_0)\in\mathbb{R}^d\times\mathbb{R}^+.$$

Proof. Function g is infinitely differentiable, and thus it must be continuous. Thus, functions $g(\boldsymbol{a}\cdot\boldsymbol{x}_i+b)$ and $g(\|\boldsymbol{x}_i-\boldsymbol{a}\|_2/b)$ $(i=1,2,\ldots,N)$ are continuous with variables $(\boldsymbol{a},b)$. □

Quite similar to Theorem 1, the approximation capability in the training set is declared as follows.

Theorem 2. *Given any infinitely differentiable function $g:\mathbb{R}\to\mathbb{R}$ which meets the conditions of Lemma 6 for additive nodes or meets the conditions of Lemma 8 for RBF nodes, for any random function sequence $\{g_n\}$ and any training set $\{(\boldsymbol{x}_j,y_j)|1\leqslant j\leqslant N\}$, let $\boldsymbol{y}=(y_1,y_2,\ldots,y_N)^{\mathrm{T}}$, $\boldsymbol{g}_n=(g_n(\boldsymbol{x}_1),g_n(\boldsymbol{x}_2),\ldots,g_n(\boldsymbol{x}_N))^{\mathrm{T}}$ and $\boldsymbol{f}_0=(0,0,\ldots,0)^{\mathrm{T}}$, and then $\lim_{n\to\infty}\|\boldsymbol{f}_n-\boldsymbol{y}\|_2=0$ holds with probability of 1 if sequence $\{\boldsymbol{f}_n\}$ generates according to (4) and (5).*

Proof. At first, we note that the conclusion of Lemma 5 still holds if we use the vectors in $\mathbb{R}^N$ instead of the functions in $L^2(X)$, and Lemma 2 and Lemma 3 can be replaced by Lemma 6 and Lemma 8, respectively. Then all the discussion on $L^2(X)$ in Theorem 1 can be ported to another complete space $\mathbb{R}^N$. Thus, a version of Theorem 1 based on $\mathbb{R}^N$ is still true as stated in Theorem 2. □

5 Performance Evaluation

In this section, the performance of the proposed LCI-ELM is compared with that of I-ELM, CI-ELM, EI-ELM, EM-ELM and EIR-ELM with respect to benchmark regression problems. The source codes are provided②.

5.1 Datasets and Environment Settings

All the problems studied in the experiments can be divided into two kinds: 1) approximation of function $SinC(x)$ and 2) 10 real-world regression problems. The training set and the testing set in SinC case are both randomly generated on the interval $[-10, 10]$ and large uniform noise distributed in $[-0.2, 0.2]$ has been added to all the training samples while the testing data remains noise-free. For real-world regression problems, the Bodyfat and Space-ga datasets are taken

②http://wuc.scse.hebut.edu.cn/dliu/lcielm.htm, Mar. 2017.

from Statlib[3] while other datasets are taken from the UCI database[4][5]. The training samples and the testing samples of each case are randomly selected from the original dataset. In the experiments, all the inputs are normalized into the range $[-1, 1]$ and the outputs are normalized into $[0, 1]$.

As in our previous analysis, additive hidden nodes and RBF hidden nodes are used to construct networks in the experiments. Sigmoid function $G(\boldsymbol{x}, \boldsymbol{a}, b) = 1/(1 + \exp(\boldsymbol{a} \cdot \boldsymbol{x} - b))$ is used as the activation function for additive nodes; the input weight $\boldsymbol{a}$ and the hidden bias b are randomly chosen from the range $[-1, 1]$. For RBF nodes, the activation function is the Gaussian function $G(\boldsymbol{x}, \boldsymbol{a}, b) = \exp(-b\|\boldsymbol{x} - \boldsymbol{a}\|_2^2)$. The centre $\boldsymbol{a}$ is randomly chosen from the range $[-1, 1]$, whereas the impact factor b is randomly chosen from $[0, 0.5]$.

Two parameters (k, λ) are to be specified by the users. Generally speaking, we can choose a larger k and a smaller λ to achieve a higher convergence rate, but the testing error may be more likely to oscillate. Thus the parameters need to be chosen appropriately. Instead of trying a wide range of k and λ, we propose a heuristic strategy to give the values of these parameters. The parameter k determines the number of the hidden nodes of the initial network. To start learning with a relatively small size network, the initial hidden layer is restricted to have no more than 15 nodes. To make it simpler, we construct the initial network with only 5, 10 or 15 hidden nodes. That is to say, the value of k is chosen from $\{4, 9, 14\}$. The parameter λ determines how fast the number of newly added hidden nodes decreases. We choose $\lambda = 0.02$ so that the network would have approximately 150 to 1 000 hidden nodes before the incremental number of hidden nodes decreases to 1 when $k = 4$, 9 or 14. The experimental results shown below are all on condition that the number of hidden nodes is in the range $[150, 1\,000]$. In summary, we set $\lambda = 0.02$ for all cases, we try $k = 4, 9, 14$ for each case, and then the best one in $\{4, 9, 14\}$ is chosen as k. Such a simple strategy can work well, too. Table 1 shows key information on the datasets and the user-specified parameters for LCI-ELM.

All the simulations are carried out in MATLAB 7.0 environment on the same PC with a Core2 2.39 GHz CPU and 2G RAM. Twenty trials have been conducted for each problem; the simulation results include the mean testing root mean square error (mean RMSE), the corresponding standard deviation (Dev.) and the average training time.

5.2 Comparison Among I-ELM, CI-ELM and LCI-ELM

We first compare the generalization performance of LCI-ELM with that of I-ELM and CI-ELM. In the simulation, the expected learning accuracy ϵ is set to be 0 for all three algorithms, thereby the number of hidden nodes reaches the maximum. In order to illustrate that LCI-ELM can obtain better generalization performance with fewer hidden nodes, we set $L_{\max} = 150$ for LCI-ELM and $L_{\max} = 300$ for I-ELM and CI-ELM. The performance of LCI-ELM, I-ELM and CI-ELM with additive and RBF nodes is shown in Table 2 and Table 3 respectively. The best results (testing RMSE) among all algorithms are shown in boldface.

Table 1. Key Information on the Datasets and the User-Specified Parameters for LCI-ELM

Dataset	Number of Training Samples	Number of Testing Samples	Number of Attributes (Inputs)	(k, λ)
SinC	1 000	500	1	(9, 0.02)
Abalone	2 000	2 177	8	(9, 0.02)
AutoMPG	200	192	7	(4, 0.02)
Bodyfat	168	84	14	(4, 0.02)
Concrete Strength	500	530	8	(9, 0.02)
Concrete Slump	70	33	7	(9, 0.02)
Energy Efficiency[6]	400	368	8	(14, 0.02)
Protein	10 000	6 000	9	(9, 0.02)
Servo	80	87	4	(4, 0.02)
Space-ga	2 071	1 036	6	(4, 0.02)
Yacht	200	108	6	(14, 0.02)

[3] http://wuc.scse.hebut.edu.cn/dliu/lcielm.htm, Mar. 2017.

[4] http://lib.stat.cmu.edu/datasets, Mar. 2017.

[5] http://archive.ics.uci.edu/ml, Mar. 2017.

[6] This dataset has two outputs and their simulation results are shown separately.

Table 2. Performance Comparison of LCI-ELM, I-ELM and CI-ELM with Additive Nodes

Dataset	I-ELM (Additive)			CI-ELM (Additive)			LCI-ELM (Additive)		
	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)
SinC	0.222 0	0.006 0	0.047 7	0.211 3	0.010 1	0.048 4	**0.011 6**	0.002 6	0.037 5
Abalone	0.088 5	0.002 9	0.095 3	0.079 8	0.001 4	0.102 3	**0.075 1**	0.000 6	0.075 0
AutoMPG	0.107 8	0.004 9	0.022 7	0.095 4	0.003 1	0.018 0	**0.087 4**	0.002 3	0.012 5
Bodyfat	0.061 2	0.012 5	0.024 2	0.031 7	0.006 1	0.021 9	**0.026 7**	0.003 4	0.013 3
Concrete Strength	0.142 1	0.005 5	0.035 9	0.129 9	0.002 8	0.035 2	**0.111 3**	0.003 5	0.025 8
Concrete Slump	0.081 1	0.010 3	0.018 0	0.075 1	0.006 3	0.016 4	**0.056 7**	0.007 8	0.006 3
Energy Efficiency 1	0.159 2	0.004 7	0.032 0	0.162 4	0.005 9	0.029 7	**0.136 9**	0.002 2	0.016 4
Energy Efficiency 2	0.173 7	0.004 0	0.032 0	0.178 7	0.005 0	0.025 0	**0.161 8**	0.002 6	0.017 2
Protein	0.256 3	0.002 7	0.436 7	0.247 4	0.001 7	0.466 4	**0.240 0**	0.001 4	0.369 5
Servo	0.168 0	0.002 2	0.016 4	0.169 1	0.004 8	0.014 1	**0.141 7**	0.006 9	0.006 3
Space-ga	0.050 9	0.003 9	0.087 5	0.048 6	0.001 7	0.079 7	**0.043 6**	0.003 3	0.063 3
Yacht	0.294 2	0.005 1	0.023 4	0.341 3	0.016 9	0.022 7	**0.179 2**	0.016 0	0.005 5

Table 3. Performance Comparison of LCI-ELM, I-ELM and CI-ELM with RBF Nodes

Dataset	I-ELM (RBF)			CI-ELM (RBF)			LCI-ELM (RBF)		
	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)
SinC	0.078 2	0.004 9	0.044 5	0.058 6	0.005 7	0.046 9	**0.021 8**	0.007 1	0.044 5
Abalone	0.089 0	0.004 3	0.159 4	0.081 0	0.001 0	0.160 2	**0.077 0**	0.001 9	0.110 9
AutoMPG	0.106 4	0.011 7	0.028 1	0.104 3	0.010 8	0.023 4	**0.091 2**	0.002 4	0.015 6
Bodyfat	0.084 4	0.020 2	0.038 3	0.070 1	0.009 7	0.032 0	**0.040 7**	0.007 2	0.028 1
Concrete Strength	0.131 3	0.006 7	0.085 9	0.128 1	0.004 5	0.068 0	**0.106 6**	0.003 0	0.028 1
Concrete Slump	0.106 4	0.014 9	0.018 0	0.090 0	0.010 1	0.018 8	**0.058 4**	0.007 9	0.007 0
Energy Efficiency 1	0.184 7	0.010 7	0.040 6	0.213 9	0.012 8	0.030 5	**0.133 4**	0.003 2	0.021 1
Energy Efficiency 2	0.197 6	0.009 5	0.033 6	0.223 3	0.015 3	0.035 9	**0.157 6**	0.002 5	0.021 9
Protein	0.255 0	0.002 2	0.878 9	0.252 7	0.002 3	0.957 8	**0.240 9**	0.001 5	0.594 5
Servo	0.180 7	0.003 2	0.019 5	0.176 9	0.005 2	0.015 6	**0.152 2**	0.008 3	0.008 6
Space-ga	0.068 2	0.017 3	0.114 1	0.046 6	0.001 4	0.082 8	**0.045 0**	0.004 2	0.065 6
Yacht	0.307 1	0.012 1	0.029 7	0.364 3	0.021 3	0.025 0	**0.159 1**	0.013 3	0.010 9

As observed from Table 2 and Table 3, the generalization performance obtained by LCI-ELM with 150 hidden nodes is better than that obtained by either I-ELM or CI-ELM with 300 hidden nodes, for all the tested cases. This demonstrates that LCI-ELM achieves a higher convergence rate. Furthermore, the training time of LCI-ELM with 150 hidden nodes is less than that of I-ELM or CI-ELM with 300 hidden nodes for all except the SinC case (RBF nodes) in which the training time of the three algorithms is close. Taking the Bodyfat case in Table 2 as an example, the mean testing RMSE is 0.026 7 for LCI-ELM with 150 hidden nodes, 0.031 7 for CI-ELM and 0.061 2 for I-ELM with 300 hidden nodes. LCI-ELM achieves a much smaller testing RMSE. Meanwhile, the training speed of LCI-ELM is almost twice that of CI-ELM or I-ELM. This means that LCI-ELM can achieve better generalization performance with more compact network architecture compared with I-ELM and CI-ELM, and LCI-ELM needs less training time.

Fig.1 and Fig.2 show the testing error updating curves of I-ELM, CI-ELM and LCI-ELM for the Abalone and Concrete Strength cases, respectively. It can be seen that the testing error of LCI-ELM decreases stably as I-ELM, and CI-ELM and LCI-ELM obviously converge much faster than I-ELM and CI-ELM. This means that the LCI-ELM needs fewer hidden nodes to obtain the same testing accuracy. In fact, similar curves have been found for other tested cases.

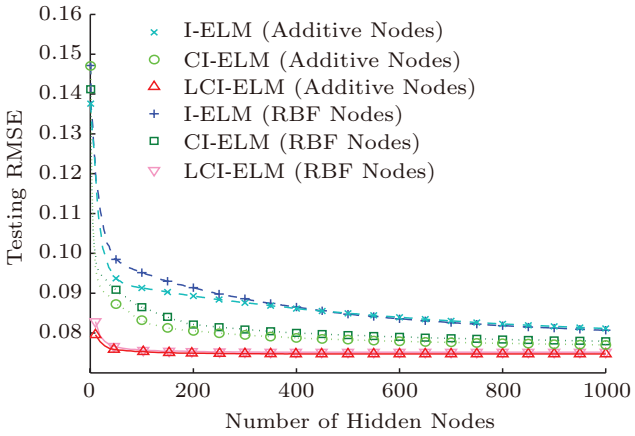

Fig.1. Testing error updating curves of I-ELM, CI-ELM and LCI-ELM for the Abalone case.

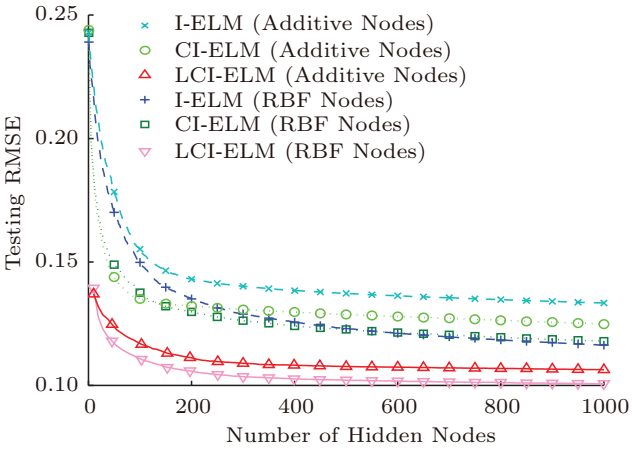

Fig.2. Testing error updating curves of I-ELM, CI-ELM and LCI-ELM for the Concrete Strength case.

5.3 Comparison Between LCI-ELM and EI-ELM

EI-ELM is an enhanced algorithm for I-ELM and it can achieve a faster convergence rate[32]. But its time cost is relatively large: the training time of EI-ELM with parameter k is around k times that of I-ELM when the same number of hidden nodes is added[32]. However, the training time of LCI-ELM is slightly longer than that of I-ELM with the same number of hidden nodes since LCI-ELM has a more complex weights calculating method. In order to compare the training time as well as the generalization performance between LCI-ELM and EI-ELM with the same number of hidden nodes, we set the same expected learning accuracy $\epsilon = 0$ and the same maximum number of hidden nodes $N_{\max} = 300$ for both LCI-ELM and EI-ELM. Table 4 and Table 5 show the performance of LCI-ELM and EI-ELM ($k = 10$, $k = 20$) with additive and RBF nodes, respectively. The apparently better results are shown in boldface.

It can be seen that the training time of LCI-ELM is much shorter than that of EI-ELM ($k = 10$, $k = 20$) for all the tested cases. Meanwhile, LCI-ELM achieves better generalization performance in all the datasets except the Bodyfat and Space-ga cases with RBF hidden nodes. In the Yacht case in Table 5, for example, the testing RMSE of LCI-ELM, which is 0.146 2, is much smaller than that of EI-ELM ($k = 10$) which is 0.252 8 and that of EI-ELM ($k = 20$) which is 0.242 9. In addition, the training speed of LCI-ELM is almost 8 and 12 times higher than that of EI-ELM ($k = 10$) and EI-ELM ($k = 20$), respectively. This means that LCI-ELM can achieve similar or better generalization performance with the same number of hidden nodes compared with EI-ELM ($k = 10$, $k = 20$). But the training process of LCI-ELM is much shorter. We also present the testing error updating curves of LCI-ELM and EI-ELM ($k = 10$, $k = 20$) for the Abalone and Concrete Strength cases in Fig.3 and Fig.4 respectively. We can see that LCI-ELM with additive nodes and RBF nodes obviously converges faster than EI-ELM ($k = 10$, $k = 20$) with additive nodes and RBF nodes, respectively.

5.4 Comparison Among EM-ELM, EIR-ELM and LCI-ELM

Similar to the proposed algorithm, LCI-ELM, EM-ELM allows adding one or more hidden nodes into the network at one time. EM-ELM aims to obtain minimal training error at each training step, which makes it different from LCI-ELM. EIR-ELM[33] is an improved

Table 4. Performance Comparison of LCI-ELM and EI-ELM ($k = 10$, $k = 20$) with Additive Nodes

Dataset	EI-ELM ($k = 10$, Additive)			EI-ELM ($k = 20$, Additive)			LCI-ELM (Additive)		
	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)
SinC	0.170 4	0.003 9	0.403 9	0.152 0	0.003 5	0.792 2	**0.011 3**	0.003 4	0.073 4
Abalone	0.078 8	0.000 7	0.796 9	0.077 8	0.000 6	1.394 5	**0.074 9**	0.000 3	0.160 2
AutoMPG	0.095 8	0.003 2	0.143 0	0.094 4	0.002 8	0.274 2	**0.087 5**	0.002 7	0.037 5
Bodyfat	0.028 3	0.005 4	0.135 2	0.026 1	0.005 4	0.263 3	**0.025 1**	0.003 3	0.032 8
Concrete Strength	0.128 5	0.001 8	0.255 5	0.125 4	0.001 7	0.481 3	**0.108 8**	0.003 2	0.043 8
Concrete Slump	0.070 9	0.004 4	0.096 9	0.068 6	0.003 5	0.178 9	**0.050 6**	0.010 5	0.012 5
Energy Efficiency 1	0.148 0	0.002 3	0.214 1	0.144 7	0.002 0	0.417 2	**0.136 3**	0.003 0	0.032 8
Energy Efficiency 2	0.167 0	0.002 1	0.218 8	0.166 0	0.002 2	0.418 0	**0.158 7**	0.003 2	0.034 4
Protein	0.246 9	0.000 7	3.621 1	0.245 5	0.000 4	6.718 8	**0.238 3**	0.000 8	0.664 8
Servo	0.157 6	0.001 8	0.090 6	0.151 8	0.002 7	0.182 8	**0.137 9**	0.006 2	0.028 1
Space-ga	0.047 6	0.002 1	0.710 9	0.047 9	0.001 7	1.382 8	**0.042 8**	0.001 9	0.164 8
Yacht	0.287 7	0.008 0	0.143 0	0.278 4	0.008 4	0.271 9	**0.161 2**	0.009 1	0.018 0

Table 5. Performance Comparison of LCI-ELM and EI-ELM ($k = 10$, $k = 20$) with RBF Nodes

Dataset	EI-ELM ($k = 10$, RBF)			EI-ELM ($k = 20$, RBF)			LCI-ELM (RBF)		
	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)	Mean	Dev.	Time (s)
SinC	0.055 8	0.001 0	0.356 3	0.056 7	0.000 6	0.711 7	**0.022 0**	0.005 6	0.078 1
Abalone	0.078 2	0.000 5	1.439 8	0.077 2	0.000 4	2.525 0	**0.075 1**	0.000 5	0.171 1
AutoMPG	0.094 0	0.002 8	0.189 1	0.091 7	0.002 1	0.319 5	**0.090 9**	0.002 1	0.035 2
Bodyfat	0.042 5	0.004 4	0.219 5	**0.039 2**	0.002 9	0.338 3	0.041 1	0.008 7	0.039 1
Concrete Strength	0.110 6	0.001 5	0.359 8	0.107 4	0.001 8	0.589 8	**0.104 8**	0.002 1	0.059 4
Concrete Slump	0.067 7	0.004 4	0.096 9	0.061 1	0.004 5	0.211 7	**0.050 3**	0.005 9	0.015 6
Energy Efficiency 1	0.156 4	0.005 2	0.253 9	0.152 9	0.004 1	0.478 1	**0.129 4**	0.001 9	0.043 0
Energy Efficiency 2	0.177 7	0.005 9	0.291 4	0.174 9	0.003 8	0.477 3	**0.154 0**	0.002 3	0.043 0
Protein	0.246 7	0.000 6	7.444 5	0.245 7	0.000 4	13.171 1	**0.239 9**	0.002 2	1.171 9
Servo	0.169 7	0.002 2	0.099 2	0.166 3	0.001 6	0.207 0	**0.152 0**	0.007 4	0.028 9
Space-ga	0.045 0	0.001 2	0.745 3	**0.043 9**	0.000 4	2.039 1	0.046 3	0.002 9	0.198 4
Yacht	0.252 8	0.005 1	0.203 9	0.242 9	0.007 6	0.303 9	**0.146 2**	0.008 1	0.022 7

version of EM-ELM based on the regularization method which aims to deal with the over-fitting problem. We compare the testing RMSE of the three algorithms (LCI-ELM, EM-ELM and EIR-ELM with parameter $k = 5$) at each step when a chunk of hidden nodes are added. In order to achieve equitable comparisons, an equal number of hidden nodes (calculated using (5)) are added at each step for LCI-ELM and EM-ELM, and parameters (k and λ) are configured according to Table 1. In [33], the authors declared it is not efficient for EIR-ELM to add new hidden nodes to existing network group by group. Therefore, we add hidden nodes one by one for EIR-ELM, and the initial number of hidden nodes is set to be the same with those of the other two algorithms. There is a regularization parameter C to be specified for EIR-ELM. We follow the same way in [33] to choose C, that is, C takes a proper value in the set $\{2^{-20}, 2^{-19}, \ldots, 2^{0}, \ldots, 2^{19}, 2^{20}\}$ for each dataset.

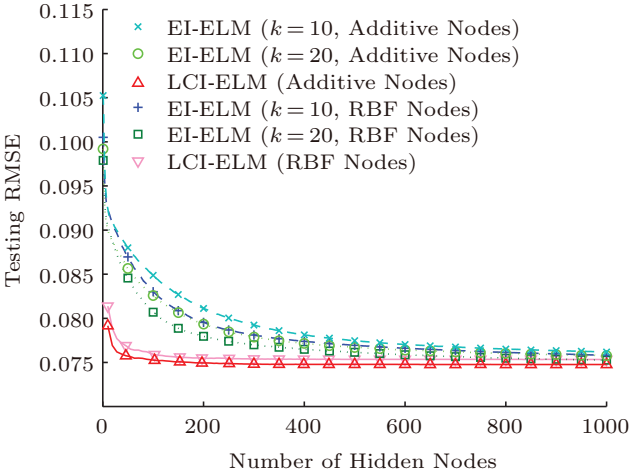

Fig.3. Testing error updating curves of EI-ELM ($k = 10$, $k = 20$) and LCI-ELM for the Abalone case.

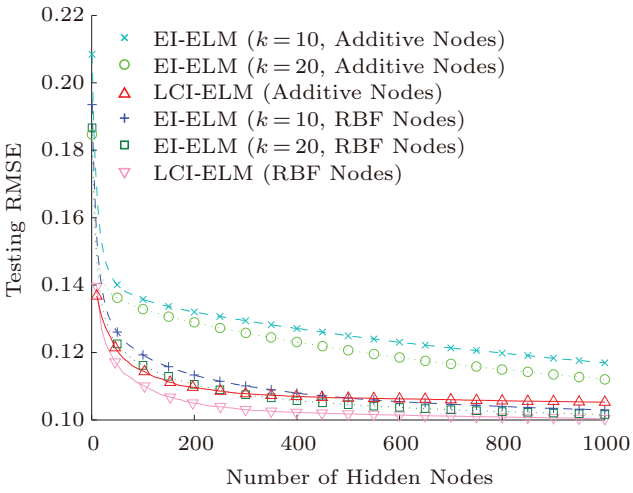

Fig.4. Testing error updating curves of EI-ELM ($k = 10$, $k = 20$) and LCI-ELM for the Concrete Strength case.

As we know, the lower the training error is, the higher the over-fitting risk is. In this subsection, we investigate the over-fitting risk among EM-ELM, EIR-ELM and LCI-ELM. We also take the Abalone and the Concrete Strength cases as instances for comparison as before. The results are shown in Fig.5 and Fig.6. From them, EM-ELM shows lower testing errors when there are only a small amount of hidden nodes in the network. EM-ELM may benefit from the error minimization strategy and has a higher convergence rate of generalization error than LCI-ELM at the beginning of training. But after a turning point (approximately 30 hidden nodes for the Abalone case and 80 hidden nodes for the Concrete Strength case), the generalization error of EM-ELM starts to rise steeply with more hidden nodes added. Meanwhile, EM-ELM is designed to decrease the training error monotonously

in the whole training process. This means EM-ELM encounters the over-fitting problem. EIR-ELM shows better performance than EM-ELM and its RMSE curves keep smooth in a wider range. However, the generalization error of EIR-ELM has an upward swing when there are more hidden nodes (approximately 60 hidden nodes for the Abalone case and 120 hidden nodes for the Concrete Strength case). Although EIR-ELM can relieve the over-fitting problem in some degree, the problem is not well solved. Especially, EM-ELM and EIR-ELM will have unacceptably bad performance when the number of the hidden nodes exceeds a certain range. However, LCI-ELM does not suffer from this problem in these cases. That is to say, over-fitting does not happen easily for LCI-ELM while adding hidden nodes.

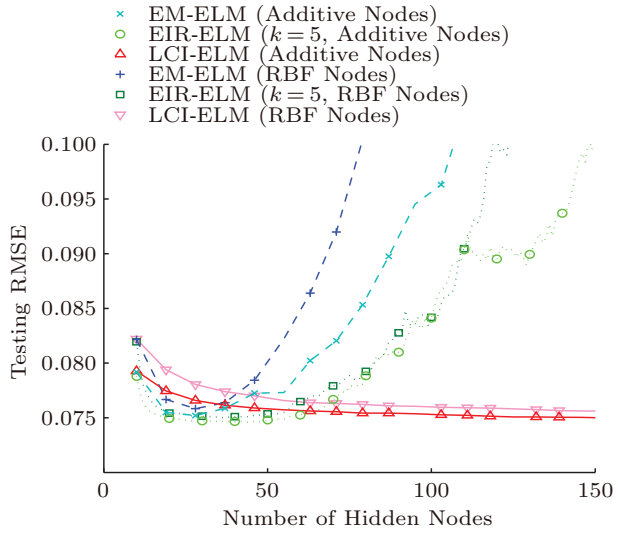

Fig.5. Testing error updating curves of EM-ELM, EIR-ELM ($k = 5$) and LCI-ELM for the Abalone case.

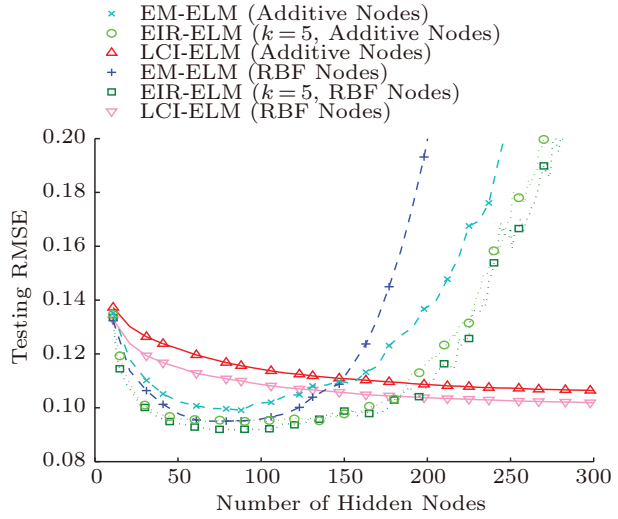

Fig.6. Testing error updating curves of EM-ELM, EIR-ELM ($k = 5$) and LCI-ELM for the Concrete Strength case.

6 Conclusions

This paper proposed a length-changeable incremental ELM (LCI-ELM) based on I-ELM for regression tasks. Different from other incremental extreme learning machines, such as I-ELM, CI-ELM and EI-ELM, LCI-ELM allows more than one hidden node to be added into the network at a time and the number of newly added nodes each time is changeable. Meanwhile, LCI-ELM retains the I-ELM's simplicity in calculating the output weights.

Like the theoretical analysis in [21], this paper proved the universal approximation capability under the meaning of L^2 norm. The proof is similar to that in [21] and the requirements for the activation functions are the same with those in [21]. The condition $e_n \perp (e_{n-1} - e_n)$ which is important in the proof in [21], however, is not required in our proof. It is worth mentioning that the same method can be used to demonstrate the approximation in the training set. The main difference between these two types of approximation is the discourse domain: one is function space $L^2(X)$ and the other is Euclidean space $\mathbb{R}^N$. What is more, there are no restrictions on the number of newly added hidden nodes, thereby the proof is still valid when the hidden-node-adding strategy alters.

Experimental results showed that LCI-ELM achieves good performance even if the parameters are chosen simply: $\lambda = 0.02$ and k from 4, 9 and 14. LCI-ELM converges much faster than I-ELM and CI-ELM. The training time of LCI-ELM is much shorter than that of EI-ELM ($k = 10$ and $k = 20$) for networks of the same size and LCI-ELM usually achieves a faster convergence rate. Compared with EM-ELM and EIR-ELM, LCI-ELM can converge more stably: the testing error decreases while new hidden nodes are added successively. Over-fitting does not usually occur for LCI-ELM.

In this paper, the number of newly added nodes at each step is set to decrease and there are two control parameters: k and λ. In the experiments, the two parameters are chosen in a quite simple way. Although the method is somewhat effective, how to choose the parameters properly for different datasets to achieve better performance is still a problem requiring further investigation. Proposals for new ways of adding hidden nodes are also worth investigating in the future.

References

[1] Castano A, Fernández-Navarro F, Hervás Martinez C. PCA-ELM: A robust and pruned extreme learning machine

approach based on principal component analysis. *Neural Processing Letters*, 2013, 37(3): 377-392.

[2] Chen H, Gong Y, Hong X. Online modeling with tunable RBF network. *IEEE Transactions on Cybernetics*, 2013, 43(3): 935-947.

[3] Frénay B, Verleysen M. Using SVMs with randomised feature spaces: An extreme learning approach. In *Proc. the 18th European Symposium on Artificial Neural Networks*, Apr. 2010, pp.315-320.

[4] Shin Y, Ghosh J. Approximation of multivariate functions using ridge polynomial networks. In *Proc. International Joint Conference on Neural Networks*, June 1992, pp.380-385.

[5] Park B J, Kim W D, Oh S K, Pedrycz W. Fuzzy set-oriented neural networks based on fuzzy polynomial inference and dynamic genetic optimization. *Knowledge and Information Systems*, 2014, 39(1): 207-240.

[6] Han F, Huang D S. Improved extreme learning machine for function approximation by encoding a priori information. *Neurocomputing*, 2006, 69(16/17/18): 2369-2373.

[7] Lin F J, Hung Y C, Ruan K C. An intelligent second-order sliding-mode control for an electric power steering system using a wavelet fuzzy neural network. *IEEE Transactions on Fuzzy Systems*, 2014, 22(6): 1598-1611.

[8] Capizzi G, Capizzi C, Bonanno F. Innovative second-generation wavelets construction with recurrent neural networks for solar radiation forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 2012, 23(11): 1805-1815.

[9] Hornik K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 1991, 4(2): 251-257.

[10] Leshno M, Lin V Y, Pinkus A, Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 1993, 6(6): 861-867.

[11] Park J, Sandberg I W. Universal approximation using radial-basis-function networks. *Neural Computation*, 1991, 3(2): 246-257.

[12] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: Theory and applications. *Neurocomputing*, 2006, 70(1/2/3): 489-501.

[13] Huang G B, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, Cybernetics, Part B* (*Cybernetics*), 2012, 42(2): 513-529.

[14] Wang S J, Chen H L, Yan W J, Chen Y H, Fu X L. Face recognition and micro-expression recognition based on discriminant tensor subspace analysis plus extreme learning machine. *Neural Processing Letters*, 2014, 39(1): 25-43.

[15] Liu D, Wu Y, Jiang H. FP-ELM: An online sequential learning algorithm for dealing with concept drift. *Neurocomputing*, 2016, 207(26): 322-334.

[16] Han D H, Zhang X, Wang G R. Classifying uncertain and evolving data streams with distributed extreme learning machine. *Journal of Computer Science and Technology*, 2015, 30(4): 874-887.

[17] Zhang T, Dai Q, Ma Z. Extreme learning machines' ensemble selection with GRASP. *Applied Intelligence*, 2015, 43(2): 439-459.

[18] Nie L, Jiang H, Ren Z *et al.* Query expansion based on crowd knowledge for code search. *IEEE Transactions on Services Computing*, 2016, 9(5): 771-783.

[19] Deng C W, Huang G B, Xu J *et al.* Extreme learning machines: New trends and applications. *Science China Information Sciences*, 2015, 58(2): 1-16.

[20] Jiang H, Nie L, Sun Z *et al.* ROSF: Leveraging information retrieval and supervised learning for recommending code snippets. *IEEE Transactions on Services Computing*, 2016. doi:10.1109/TSC.2016.2592909

[21] Huang G B, Chen L, Siew C K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 2006, 17(4): 879-892.

[22] Wang N, Han M, Dong N, Er M J. Constructive multi-output extreme learning machine with application to large tanker motion dynamics identification. *Neurocomputing*, 2014, 128: 59-72.

[23] Feng G, Huang G B, Lin Q, Gay R. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 2009, 20(8): 1352-1357.

[24] Wang N, Er M J, Han M. Parsimonious extreme learning machine using recursive orthogonal least squares. *IEEE Transactions on Neural Networks and Learning Systems*, 2014, 25(10): 1828-1841.

[25] Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A. OP-ELM: Optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, 2010, 21(1): 158-162.

[26] Luo X, Liu F, Yang S, Wang X, Zhou Z. Joint sparse regularization based sparse semi-supervised extreme learning machine (S3ELM) for classification. *Knowledge-Based Systems*, 2015, 73: 149-160.

[27] Zhang R, Lan Y, Huang G B, Xu Z B. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. *IEEE Transactions on Neural Networks and Learning Systems*, 2012, 23(2): 365-371.

[28] Zhang R, Lan Y, Huang G B, Xu Z B, Soh Y C. Dynamic extreme learning machine and its approximation capability. *IEEE Transactions on Cybernetics*, 2013, 43(6): 2054-2065.

[29] Feng G, Lan Y, Zhang X *et al.* Dynamic adjustment of hidden node parameters for extreme learning machine. *IEEE Transactions on Cybernetics*, 2015, 45(2): 279-288.

[30] Yang Y, Wu Q M J. Extreme learning machine with subnetwork hidden nodes for regression and classification. *IEEE Transactions on Cybernetics*, 2016, 46(12): 2885-2898.

[31] Huang G B, Chen L. Convex incremental extreme learning machine. *Neurocomputing*, 2007, 70(16/17/18): 3056-3062.

[32] Huang G B, Chen L. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 2008, 71(16/17/18): 3460-3468.

[33] Xu Z, Yao M, Wu Z, Dai W. Incremental regularized extreme learning machine and it's enhancement. *Neurocomputing*, 2016, 174: 134-142.

[34] Kolmogorov A N, Fomin S V. Elements of the Theory of Functions and Functional Analysis: Measure. Graylock Press, 1961.

[35] Kwok T Y, Yeung D Y. Objective functions for training new hidden units in constructive neural networks. *IEEE Transactions on Neural Networks*, 1997, 8(5): 1131-1148.

[36] Micchelli C A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 1986, 2: 11-22.

You-Xi Wu received his Ph.D. degree in theory and new technology of electrical engineering from Hebei University of Technology, Tianjin, in 2007. He is currently a Ph.D. supervisor and a professor with Hebei University of Technology, Tianjin. His current research interests include data mining and machine learning. Dr. Wu is a senior member of CCF.

Dong Liu received his Master's degree in computer science and technology from Hebei University of Technology, Tianjin, in 2016. Now, he is a Ph.D. candidate of software engineering at Dalian University of Technology, Dalian. His research interests include machine learning and its applications in software engineering.

He Jiang received his Ph.D. degree in computer science from University of Science and Technology of China, Hefei, in 2004. He is currently a Ph.D. supervisor and a professor with Dalian University of Technology, Dalian. His current research interests include search-based software engineering and mining software repositories. Dr. Jiang is a senior member of CCF.