An Improved Local Coupled Extreme Learning Machine

Chong Liu[1], Bing Qiang Wang[2*], Xiao Lan Wang[2], Yu Lin He[3*], Rana Aamir Raza Ashfaq[3, 4]

[1] Office of Scientific Research and Practical Training, Cangzhou Technical College, Cangzhou 061001, China.
[2] Department of Information Engineering, Cangzhou Technical College, Cangzhou 061001, China.
[3] College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China.
[4] Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan.

* Corresponding authors. Emails: bingqiangwang@126.com, yulinhe@szu.edu.cn
Manuscript submitted April 15, 2016; accepted June 20, 2016.
doi: 10.17706/jsw.11.8.745-755

Abstract: Local Coupled Extreme Learning Machine (LCELM) is a recently-proposed variant of ELM, which assigns an *address* for each hidden-layer node and activates the hidden-layer node when its activated degree is less than a given threshold. In this paper, an improved version of LCELM is proposed by developing a new way to initialize the address for each hidden-layer node and calculating the activated degree of hidden-layer node with Gaussian kernel. The experimental comparison with ELM and LCELM demonstrates the feasibility and effectiveness of improve LCELM which obtains the higher testing accuracy without significantly increasing the training time of ELM.

Key words: Extreme learning machine, address of hidden-layer node, window function, Gaussian kernel.

1. Introduction

Extreme Learning Machine (ELM) [1], [2] is a fast and simple training algorithm for the Single hidden-Layer Feed-forward neural Network (SLFN). ELM randomly selects the input-layer weights and hidden-layer biases and analytically calculates the output-layer weights. Due to the simple implementation with popular programming language and good generalizations for classification [3] and clustering [4] problems, ELM has found many applications in various areas, such as face recognition [5], time series analysis [6], and unsupervised feature learning [7], etc. Up to now, many variants of ELM have been proposed to improve its stability, robustness, and generalization capability [8], [9].

Most of these improvements focus on the optimization of input-layer weights or calculation of generalized inverse of hidden-layer output matrix. Unlike the existing works, a Local Coupled Extreme Learning Machine (LCELM) [10] was recently proposed to conditionally activate the hidden-layer node by assigning an *address* for each hidden-layer node. LCELM is an extended version of Local Coupled Feedforward Neural Network (LCFNN) [11]. The main difference between LCFNN and LCELM is that the former uses the iterative way to adjust the network weights and the latter does not need any iterative adjustment to weights. Thus, LCELM has the faster training speed. Meanwhile, the universal approximation property of LCELM is also proved. However, the further analysis of LCELM shows that there are two shortcomings for LCELM: one is to assign the address for hidden-layer node in a random way and another is the improper calculation to the activated degree. These two shortcomings limit the effect of address of hidden-layer node in LCELM.

In this paper, we propose an improved LCELM to overcome these two shortcomings of LCELM. The improved LCELM uses the output of hidden-layer node as its address so as that the address is related to input and input-layer weights. In addition, we use Gaussian kernel as the window function of LCELM. The activated degree calculated with the newly-used address and window function can effectively reflect the functionality and usability of hidden-layer node. The experimental comparison with ELM [1] and LCELM [10] demonstrate the feasibility and effectiveness of improved LCELM which obtains the higher testing accuracies on the employed UCI benchmark data sets [12].

The rest of the paper is organized as follows: In Section 2, we summarize the existing LCELM. In Section 3, we give the analysis to LCELM and show the shortcomings of LCELM. Section 4 depicts the improved LCELM. Section 5 presents the experimental validation. Finally, we conclude this work with some remarks in the last section.

2. LCELM

Given the training data set $T=\left\{\left(\vec{x}_n,\vec{y}_n\right)\middle|\vec{x}_n=\left(x_{n1},x_{n2},\cdots,x_{nD}\right),\vec{y}_n=\left(y_{n1},y_{n2},\cdots,y_{nM}\right),n=1,2,\cdots,N\right\}$, where N is the number of instances in T, D is the number of conditional attributes, and M is the number of classes, $x_{nd}\in R\left(d=1,2,\cdots,D\right)$, and $y_{nm}=\begin{cases}1 \quad , \quad \text{if } \vec{x}_n \text{ belongs to the } m\text{-th } \left(m=1,2,\cdots,M\right) \text{ class} \\ 0 \quad , \quad \text{otherwise}\end{cases}$, ELM [1] randomly assigns the input-layer weight matrix and hidden-layer bias vector

$$\mathrm{W}_{D\times L}=\left[\vec{w}_1,\vec{w}_2,\cdots,\vec{w}_L\right]=\begin{bmatrix} w_{11} & w_{21} & \cdots & w_{L1} \\ w_{12} & w_{22} & \cdots & w_{L2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1D} & w_{2D} & \cdots & w_{LD} \end{bmatrix} \text{and } \mathrm{b}=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_L \end{bmatrix},$$

and analytically calculates the output-layer weight matrix $\beta_{L\times M}=\begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1M} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{L1} & \beta_{L2} & \cdots & \beta_{LM} \end{bmatrix}$ according to Eq. (1) as

$$\beta_{\mathrm{ELM}}=\mathrm{H}^{\dagger}\mathrm{Y}, \tag{1}$$

where $H^{\dagger}$ is the generalized inverse of hidden-layer output matrix $\mathrm{H}_{N\times L}=\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1L} \\ h_{21} & h_{22} & \cdots & h_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{NL} \end{bmatrix}$,

$$h_{nl}=g\left(\sum_{d=1}^{D}x_{nd}w_{ld}+b_l\right) \tag{2}$$

is the output of the l-th$\left(l=1,2,\cdots,L\right)$ hidden-layer node corresponding to the n-th$\left(n=1,2,\cdots,N\right)$ training instance, $g\left(u\right)=\dfrac{1}{1+\exp\left(-u\right)}$, $u\in\left(-\infty,+\infty\right)$ is the sigmoid activation function, $\mathrm{Y}_{N\times M}=\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1M} \\ y_{21} & y_{22} & \cdots & y_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NM} \end{bmatrix}$ is the instance output matrix, and L is the number of hidden-layer nodes.

LCELM [10] modifies the calculation of hidden-layer output matrix of ELM and gives the following Eq. (3) as

$$\bar{H}_{N\times L}=\begin{bmatrix} h_{11}\bar{f}_{11} & h_{12}\bar{f}_{12} & \cdots & h_{1L}\bar{f}_{1L} \\ h_{21}\bar{f}_{21} & h_{22}\bar{f}_{22} & \cdots & h_{2L}\bar{f}_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1}\bar{f}_{N1} & h_{N2}\bar{f}_{N2} & \cdots & h_{NL}\bar{f}_{NL} \end{bmatrix}, \tag{3}$$

where $\bar{f}_{nl}=\bar{f}\left(\vec{x}_n,\vec{a}_l\right)$ is the window function of LCELM, which is used to measure the activated degree of the l-th hidden-layer node by calculating the similarity between the n-th instance $\vec{x}_n$ and address $\vec{a}_l$ of the l-th hidden-layer node. For LCELM, the address $\vec{a}_l=\left(a_{l1},a_{l2},\cdots,a_{lD}\right)$ is also randomly assigned and f_{nl} is calculated as

$$\bar{f}_{nl}=\bar{f}\left(\vec{x}_n,\vec{a}_l\right)=F\left[S\left(\vec{x}_n,\vec{a}_l\right)\right]=\frac{2}{1+\exp\left[\dfrac{\dfrac{\theta}{\left\|\vec{x}_n-\vec{a}_l\right\|}\sin\left(\dfrac{\left\|\vec{x}_n-\vec{a}_l\right\|}{\theta}\right)}{r}\right]}, \tag{4}$$

where $F\left(u\right)=\dfrac{2}{1+\exp\left(\dfrac{u}{r}\right)},u\in\left(-\infty,+\infty\right)$ is reversed sigmoid function, $S\left(\vec{x}_n,\vec{a}_l\right)=\dfrac{\theta}{\left\|\vec{x}_n-\vec{a}_l\right\|}\sin\left(\dfrac{\left\|\vec{x}_n-\vec{a}_l\right\|}{\theta}\right)$ is wave kernel, r and θ are the learning parameters. In LCELM, r=0.4 [10]. The corresponding output-layer weight of LCELM is calculated as

$$\beta_{LCELM}=\bar{H}^{\dagger}Y. \tag{5}$$

3. Analysis to LCELM

In LCELM, the address of the l-th hidden-layer node $\vec{a}_l=\left(a_{l1},a_{l2},\cdots,a_{lD}\right)$ is randomly assigned. This kind of random initialization cannot accurately measure the relationship between the instance and hidden-layer node. Specifically, we cannot know what the impact of random address on the activation of hidden-layer node is.

In order to explain this view, we give a simple experiment to show the comparison of numbers that the$l$-th hidden-layer node is not activated in LCELM and improved LCELM (as presented in Section 4). Let the n-th training instance be $\vec{x}_n$=(0.39, 0.52, 0.18, 0.63, 0.75). We randomly select the address $\vec{a}_l$ for the l-th hidden-layer node in the interval $\left[0,1\right]$. For $\vec{a}_l'$ in the improved LCELM, we assign its value as Eq. (10). When $\left\|\vec{x}_n-\vec{a}_l\right\|\geq r$ or $\left\|\vec{x}_n-\vec{a}_l'\right\|\geq r$, we think that the l-th hidden-layer node is not activated in LCELM or improved LCELM. r ranges from 0 to 2 in step of 0.01. For each r, $\left\|\vec{x}_n-\vec{a}_l\right\|$ or $\left\|\vec{x}_n-\vec{a}_l'\right\|$ is dependently calculated for 1000 times and the numbers that $\left\|\vec{x}_n-\vec{a}_l\right\|\geq r$ and $\left\|\vec{x}_n-\vec{a}_l'\right\|\geq r$ are summarized in Fig. 1. From Fig. 1, we do not clearly know what random initialization can lead to the activation for the given r. But, this experiment also reflects that the number that the hidden-layer node is activated in the improved LCELM is less than LCELM. This indicates that the hidden-layer node with address $\vec{a}_l'$ in the improved LCELM is more easily activated than LCELM with the random address.

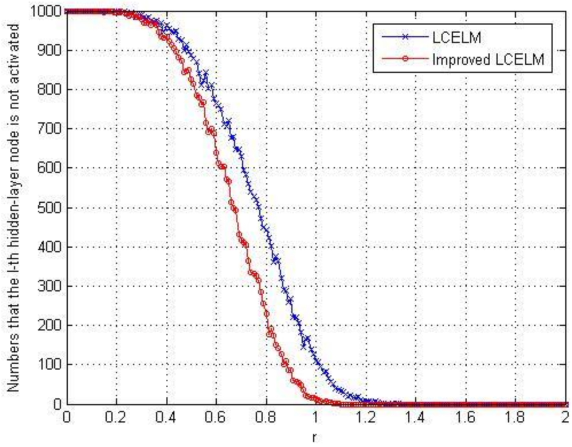


Fig. 1. The comparison of numbers that the *l*-th hidden-layer node is not activated in LCELM and improved LCELM.

In addition, Eq. (4) doesn't satisfy the conditions of window function given in [11], i.e., (1) $\bar{f}\left(\vec{x}_n,\vec{a}_l\right)\neq 1$ when $\left\|\vec{x}_n-\vec{a}_l\right\|=0$; (2) there is no such $\bar{r}$ which makes $\bar{f}\left(\vec{x}_n,\vec{a}_l\right)=0$ when $\left\|\vec{x}_n-\vec{a}_l\right\|\geq\bar{r}$; and (3) $\bar{f}\left(\vec{x}_n,\vec{a}_l\right)$ is not a monotonic function of $\left\|\vec{x}_n-\vec{a}_l\right\|$ when $\left\|\vec{x}_n-\vec{a}_l\right\|\in\left[0,\bar{r}\right)$. Regarding the monotonicity of $\bar{f}\left(\vec{x}_n,\vec{a}_l\right)$, we let

$$f\left(u\right)=\frac{2}{1+\exp\left[\frac{\frac{\theta}{u}\sin\left(\frac{u}{\theta}\right)}{r}\right]},u>0 \tag{6}$$

and check what $\bar{r}$ can make $f\left(u\right)$ monotonic in the interval $\left[0,\bar{r}\right)$. Let $f'\left(u\right)=0$, we can get

$$f'\left(u\right)=\frac{-2\exp\left[\frac{\frac{\theta}{u}\sin\left(\frac{u}{\theta}\right)}{r}\right]}{\left[1+\exp\left[\frac{\frac{\theta}{u}\sin\left(\frac{u}{\theta}\right)}{r}\right]\right]^2}\frac{\left[-\frac{\theta}{u^2}\sin\left(\frac{u}{\theta}\right)+\frac{1}{u}\cos\left(\frac{u}{\theta}\right)\right]}{r}=0, \tag{7}$$

i.e.,

$$-\frac{\theta}{u^2}\sin\left(\frac{u}{\theta}\right)+\frac{1}{u}\cos\left(\frac{u}{\theta}\right)=0. \tag{8}$$

Eq. (8) can be further simplified as

$$\tan\left(\frac{u}{\theta}\right)=\frac{u}{\theta}. \tag{9}$$

By solving Eq. (9), we can get [13]

$$\frac{u_k}{\theta}\approx\frac{\left(2k+1\right)\pi}{2}-\frac{2}{\left(2k+1\right)\pi},k=1,2,3,\cdots \tag{10}$$

When k=1, $u_1 \approx 4.5\theta$ can be calculated from Eq. (10). When $u \in (0, 4.5\theta)$, we can get $f'(u) > 0$. This indicates that $f(u)$ is a monotonically increasing function in the interval $(0, 4.5\theta)$. However, we can find that when $u > 4.5\theta$, $f(u) \neq 0$ because $f(u) > 0$ holds for any $u \in (0, +\infty)$.

4. Improved LCELM

In this section, we give our improved LCELM to overcome the above-mentioned shortcomings of LCELM. Our improvements include two parts: one is to use the address $\vec{a}'_l$ as show in Eq. (11) and another is to calculate the activated degree $\underline{f}_{nl}$ with Gaussian kernel as shown in Eq. (12):

$$\vec{a}'_l = (a'_{l1}, a'_{l2}, \cdots, a'_{lD}) = (x_{n1}w_{l1}, x_{n2}w_{l2}, \cdots, x_{nD}w_{lD}), \tag{11}$$

$$\underline{f}_{nl} = \underline{f}(\vec{x}_n, \vec{a}'_l) = \begin{cases} \exp\left(-\dfrac{\|\vec{x}_n - \vec{a}'_l\|^2}{2\sigma^2}\right) & , \quad \text{if } 0 \le \dfrac{\|\vec{x}_n - \vec{a}'_l\|}{\sigma} < \underline{r} \\ 0 & , \quad \text{if } \dfrac{\|\vec{x}_n - \vec{a}'_l\|}{\sigma} \ge \underline{r} \end{cases}. \tag{12}$$

In our study, we let $\underline{r} = 4.8$ (The value of $\underline{r}$ is not unique. Any $\underline{r} > 0$ can be used as window radium [11]). Then, the output-layer weight of improved LCELM is calculated as

$$\beta_{\text{Improved LCELM}} = \underline{H}^{\dagger}Y, \tag{13}$$

where, the hidden-layer output matrix of improved LCELM is

$$\underline{H}_{N\times L} = \begin{bmatrix} h_{11}\underline{f}_{11} & h_{12}\underline{f}_{12} & \cdots & h_{1L}\underline{f}_{1L} \\ h_{21}\underline{f}_{21} & h_{22}\underline{f}_{22} & \cdots & h_{2L}\underline{f}_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1}\underline{f}_{N1} & h_{N2}\underline{f}_{N2} & \cdots & h_{NL}\underline{f}_{NL} \end{bmatrix}. \tag{14}$$

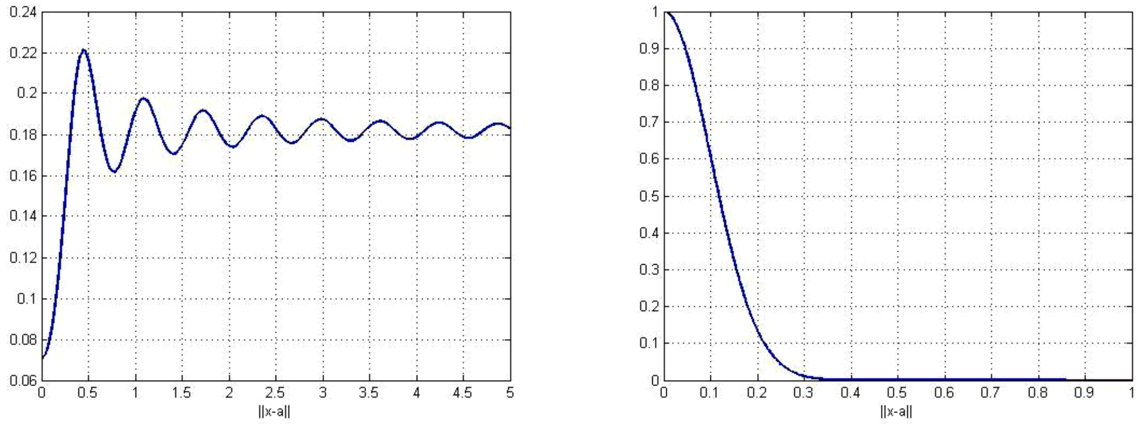
The improved LCELM calculates the output of the m-th output-layer node corresponding to the n-th training instance as

$$t_{nm} = \sum_{l=1}^{L} h_{nl}\underline{f}_{nl}\beta_{lm}. \tag{15}$$

By comparing the improved LCELM with LCELM, we can find that there are two main differences between these two learning algorithms:

1) The addresses of LCELM are fixed for any instances, while the addresses of the improved LCELM are changed with instances. It is more reasonable that the different instances correspond to the different addresses, because the fixed addresses cannot effectively distinguish the impact of the instances on the activations of hidden-layer nodes.
2) The Fig. 2 shows the differences between $\overline{f}_{nl} = \overline{f}(\vec{x}_n, \vec{a}_l)$ and $\underline{f}_{nl} = \underline{f}(\vec{x}_n, \vec{a}_l)$ when θ=r=σ=0.1. From Fig. 2, we can see that $\overline{f}_{nl} = \overline{f}(\vec{x}_n, \vec{a}_l)$ in LCELM doesn't satisfy the conditions of LCFNN window function [11] as discussed in Section 3, while $\underline{f}_{nl} = \underline{f}(\vec{x}_n, \vec{a}_l)$ strictly obeys the requirements of LCFNN window

function: (1) $\underline{f}\left(\vec{x}_n,\vec{a}_l\right)$ is a continuous function; (2) $\underline{f}\left(\vec{x}_n,\vec{a}_l\right)=1$ when $\left\|\vec{x}_n-\vec{a}_l\right\|=0$; (3) $\underline{f}\left(\vec{x}_n,\vec{a}_l\right)=0$ when $\frac{\left\|\vec{x}_n-\vec{a}_l\right\|}{\sigma}\geq \underline{r}$; and (4) $\underline{f}\left(\vec{x}_n,\vec{a}_l\right)$ is a monotonically decreasing function when $\frac{\left\|\vec{x}_n-\vec{a}_l\right\|}{\sigma}\in\left[0,\underline{r}\right)$.



(a) $\bar{f}_{nl}=\bar{f}\left(\vec{x}_n,\vec{a}_l\right)$ (b) $\underline{f}_{nl}=\underline{f}\left(\vec{x}_n,\vec{a}_l\right)$

Fig. 2. The diagrams of different window functions on LCELM and improved LCELM.

5. Experimental Validation

In this section, we carry out a series of experiments to demonstrate the feasibility and effectiveness of improved LECLM. We compare the training accuracy (Train Acc), testing accuracy (Test Acc), training time (Train Time), and testing time (Test Time) of improved LCELM with ELM [1] and LCELM [10] based on 20 UCI benchmark data sets [12]. The detailed information of data sets is summarized in Table 1. The 10-times 10-fold cross-validation procedure is used in our experiment. All experiments are implemented with Matlab 7.1 and ran on a Thinkpad X250 PC (i7-5600U 2.60GHz CPU and 8.00GB RAM) with Windows 10 OS.

Table 1. The Details of 20 UCI Data Sets

	Data sets	Attributes	Classes	Class distribution	Instances
1	Appendicitis	7	2	85/21	106
2	Auto Mpg	5	3	245/79/68	392
3	Breast Cancer	10	2	458/241	699
4	Breast CancerW-P	33	2	151/47	198
5	Cleveland	13	5	160/54/35/35/13	297
6	Credit Approval	15	2	383/307	690
7	Cylinder Bands	20	2	312/228	540
8	Ecoli	5	8	143/77/52/35/20/5/2/2	336
9	Glass	9	7	76/70/29/17/13/9/0	214
10	Haberman	3	2	225/81	306
11	Heart Disease	13	2	150/120	270
12	Iris	4	3	50×3	150
13	Libras Movement	90	15	24×15	360
14	Musk Version1	166	2	269/207	476
15	New Thyroid	5	3	150/35/30	215
16	Parkinsons	22	2	147/48	195
17	Sonar	60	2	111/97	208
18	Vowel	10	11	48×11	528
17	Vehicle	18	4	218/217/212/199	846
20	Tae	3	3	52/50/49	151

For ELM, LCELM, and improved LCELM, the number of hidden-layer nodes ranges from 10 to 200 in step of 10, i.e., $L=\{10,20,\cdots,200\}$. The learning parameters θ in Eq. (4) and σ in Eq. (12) are $\theta=\{0.1,0.2,\cdots,1.0\}$ and $\sigma=\{0.1,0.2,\cdots,1.0\}$. On each data set, we respectively select the best testing accuracies for ELM, LCELM, and improved LCELM corresponding to 20 values of L, 200 pairs of (L, θ), and 200 pairs of (L, σ). The best testing accuracy and corresponding training accuracy, training time, and testing time for each learning algorithm are listed in Table 2. From Table 2, we can see that the improved LCELM obtains the better testing accuracies on 20 selected UCI data sets than ELM and LCELM. In addition, the training time of improved LCELM isn't obviously higher than ELM and LCELM.

Table 2. Experimental Comparisons among ELM [1], LCELM [10], and Improved LCELM

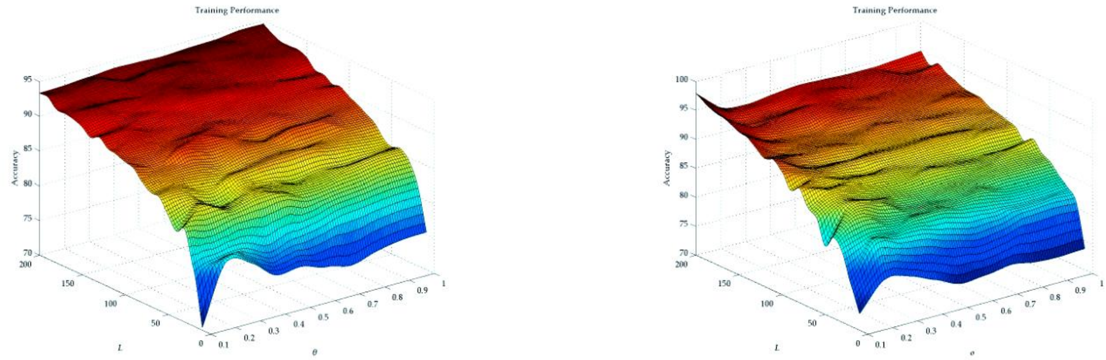
#	ELM					LCELM					Improved LCELM				
	Train Acc	Test Acc	Train Time	Test Time	L	Train Acc	Test Acc	Train Time	Test Time	(L, θ)	Train Acc	Test Acc	Train Time	Test Time	(L, σ)
1	89.6250	88.1250	0.0006	0.0001	10	90.5764	87.5000	0.0100	0.0012	(10, 0.7)	90.2569	**88.7500**	0.0102	0.0012	(10, 0.9)
2	87.9179	80.6963	0.0040	0.0002	70	82.9473	79.6839	0.0531	0.0054	(30, 0.6)	83.2328	**81.1040**	0.0463	0.0051	(20, 0.2)
3	97.3768	96.5385	0.0014	0.0001	30	98.5656	97.6303	0.2007	0.0810	(140, 1.0)	98.3607	**98.1043**	0.0633	0.2007	(140, 1.0)
4	76.2188	75.9259	0.0004	0.0000	10	90.5109	77.0492	0.0360	0.0149	(60, 0.6)	89.7810	**81.9672**	0.0273	0.0117	(40, 0.7)
5	62.4931	57.5714	0.0007	0.0000	20	66.0194	58.2418	0.0310	0.0145	(30, 0.7)	65.5340	**59.3407**	0.0291	0.0128	(30, 0.4)
6	79.9998	77.0852	0.0022	0.0001	50	81.1203	76.9231	0.1102	0.0439	(70, 0.3)	82.1577	**78.3654**	0.1229	0.0499	(80, 0.4)
7	64.6492	57.0261	0.0005	0.0000	10	68.6208	60.3205	0.0885	0.0096	(30, 0.1)	66.5782	**64.4172**	0.0587	0.0247	(30, 0.4)
8	88.2454	87.0175	0.0016	0.0001	20	87.3929	87.2835	0.0328	0.0036	(10, 1.0)	88.2976	**88.0853**	0.0363	0.0041	(10, 1.0)
9	77.4421	64.6581	0.0011	0.0000	30	76.8701	65.9402	0.0288	0.0034	(30, 0.3)	80.1487	**66.6880**	0.0349	0.0038	(40, 0.4)
10	77.7037	75.8333	0.0018	0.0001	20	77.4871	75.8332	0.0283	0.0032	(10, 1.0)	77.4147	**77.0556**	0.0324	0.0032	(10, 0.1)
11	84.4444	81.8519	0.0008	0.0000	20	94.7090	81.4815	0.0547	0.0219	(90, 0.3)	94.7092	**82.7160**	0.0578	0.0229	(90. 0.5)
12	97.1111	96.6667	0.0012	0.0001	10	98.5185	97.3333	0.0209	0.0024	(30, 0.2)	99.1852	**98.0000**	0.0314	0.0356	(40, 0.3)
13	77.4444	46.0000	0.0058	0.0002	130	55.7138	47.5556	0.0376	0.0042	(10, 0.6)	58.2929	**50.5556**	0.0518	0.0057	(20, 1.0)
14	62.1434	57.1178	0.0088	0.0002	180	68.9583	62.1108	0.0560	0.0059	(10, 0.7)	78.7403	**67.9804**	0.0197	0.0216	(50, 1.0)
15	99.8969	92.3626	0.0020	0.0001	60	96.8489	92.2674	0.0241	0.0027	(20, 0.1)	96.6454	**94.9267**	0.0248	0.0028	(20, 0.2)
16	99.8305	87.5758	0.0034	0.0002	80	98.1816	86.1111	0.0408	0.0042	(50, 1.0)	98.2486	**88.9394**	0.0415	0.0046	(50, 0.3)
17	77.4078	68.5714	0.0012	0.0001	30	90.6028	70.7143	0.0516	0.0056	(50, 0.8)	93.6418	**71.5714**	0.0773	0.0080	(70, 0.7)
18	95.9780	72.6515	0.0035	0.0002	80	97.0587	74.9241	0.1352	0.0140	(90, 0.4)	95.7438	**75.6818**	0.1556	0.0168	(90, 1.0)
19	85.9319	77.0461	0.0057	0.0002	100	83.8078	76.3437	0.2182	0.0220	(90, 0.8)	85.0404	**77.6852**	0.2530	0.0263	(100, 0.7)
20	95.8805	69.3143	0.0031	0.0001	100	96.4645	72.2571	0.0389	0.0043	(100, 0.3)	96.7565	**73.5143**	0.0518	0.0055	(120, 0.7)

The main reasons that the improved LCELM can obtain the better prediction are that the improved LCELM avoids using the random address and improper window function. The window function $\overline{f}\left(\vec{x}_n,\vec{a}_l\right)$ in Eq. (4) cannot effectively control the activation of hidden-layer node for different instances because it doesn't satisfy the requirements of window function defined in LCFNN [11]. For the instances $\vec{x}_{n_1}$ and $\vec{x}_{n_2}$ ($\vec{x}_{n_1} \neq \vec{x}_{n_2}$),

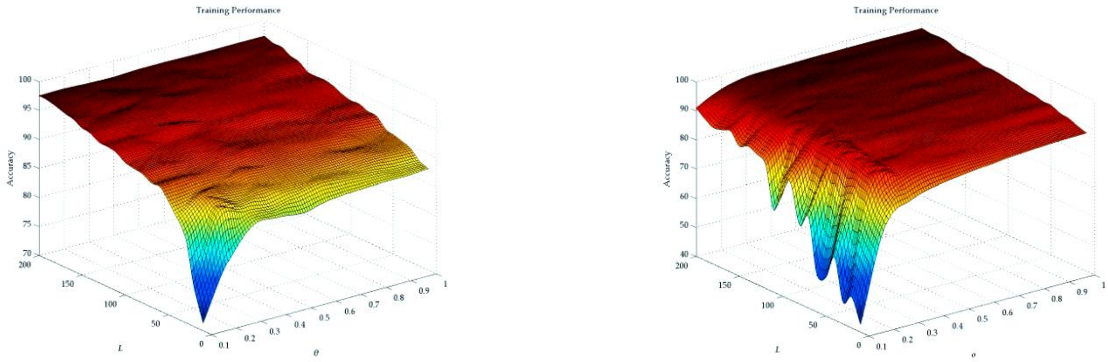
$$\overline{f}\left(\vec{x}_{n_1},\vec{a}_l\right)=\overline{f}\left(\vec{x}_{n_2},\vec{a}_l\right)$$

may hold because $f\left(u\right)$ in Eq. (6) is not a monotonic function when $u>4.5\theta$. However, the window function $\underline{f}\left(\vec{x}_n,\vec{a}_l\right)$ in Eq. (12) used in the improved LCELM can effectively deal with the activations corresponding to different instances, because $\underline{f}\left(\vec{x}_n,\vec{a}_l\right)$ is a monotonic function when $\frac{\left\|\vec{x}_n-\vec{a}_l\right\|}{\sigma}\in\left[0,\underline{r}\right)$. In addition, the random address makes LCELM more unstable than the improved LCELM. As shown in Fig. 3 and Fig. 4, we test the impacts of parameter pairs (L, θ) and (L, σ) on the learning performances of LCELM and improved LCELM, respectively. From these figures, we can find that the stability of LCELM is worse than improved LCELM.

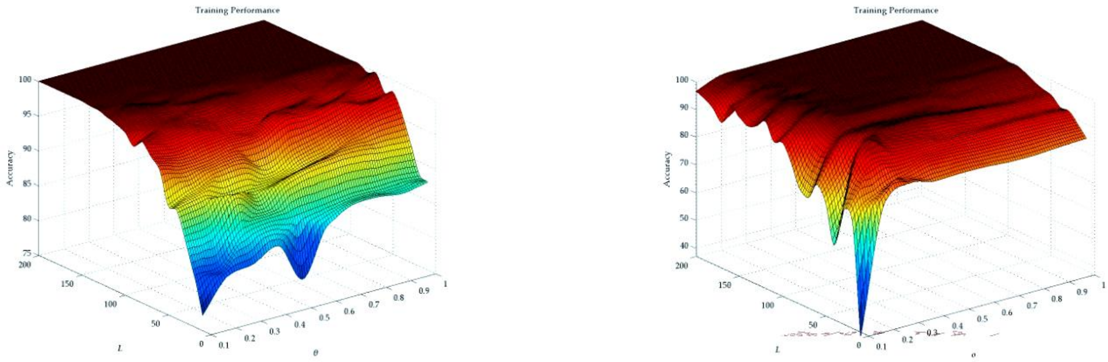
LCELM Improved LCELM



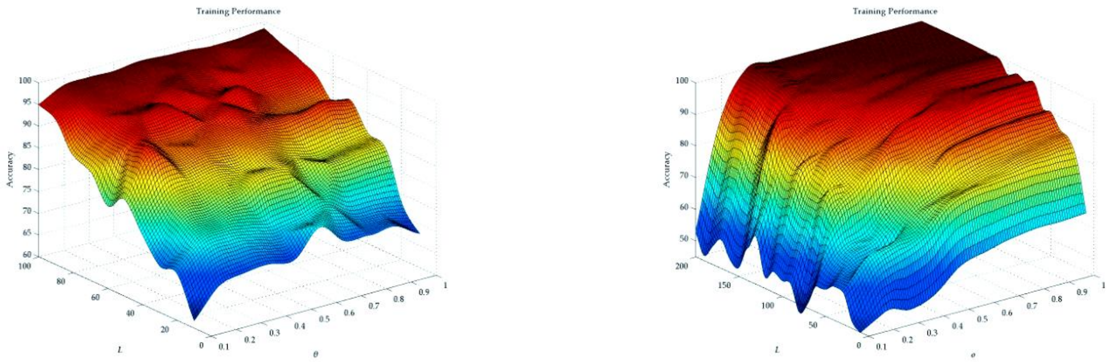
(a) On Auto Mpg data set



(b) On Ecoli data set



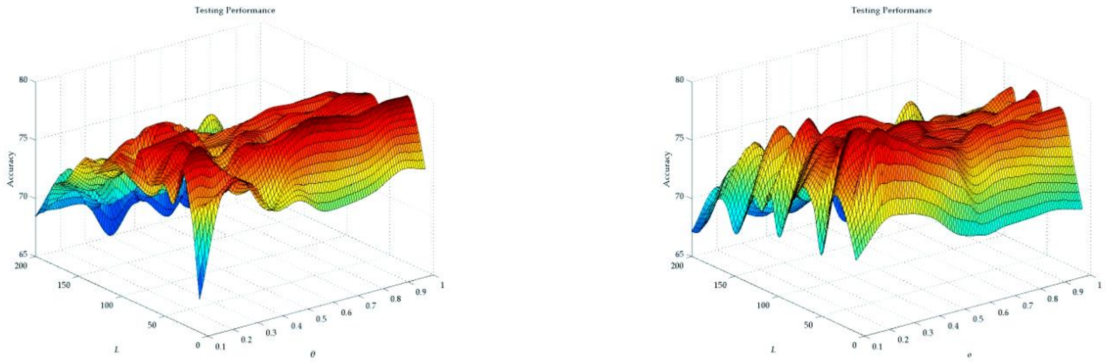
(c) On Parkinsons data set



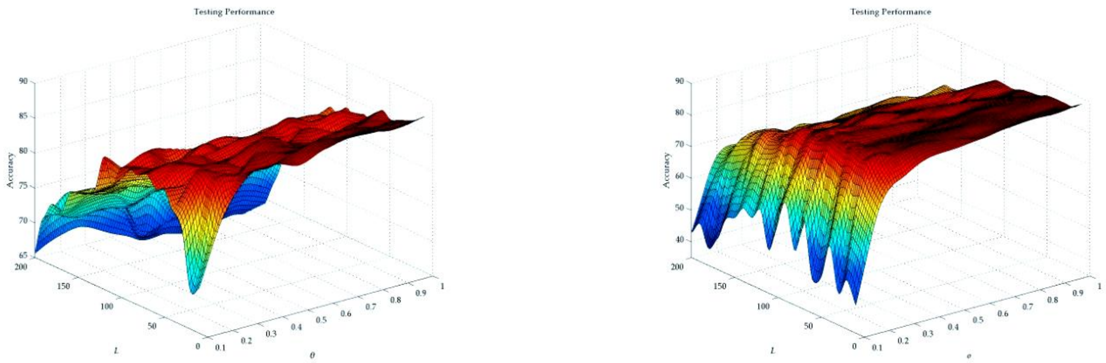
(d) On Sonar data set

Fig. 3. Impact of parameter pairs (L, θ) and (L, σ) on **training** performances of LCELM and improved LCELM.

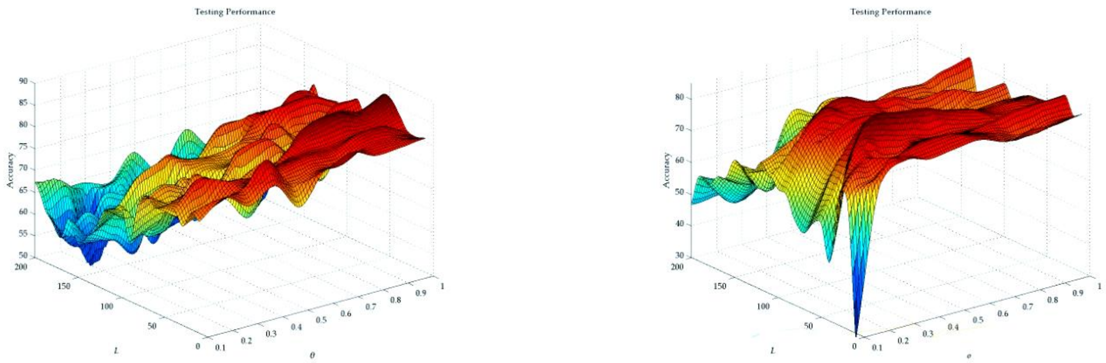
LCELM Improved LCELM



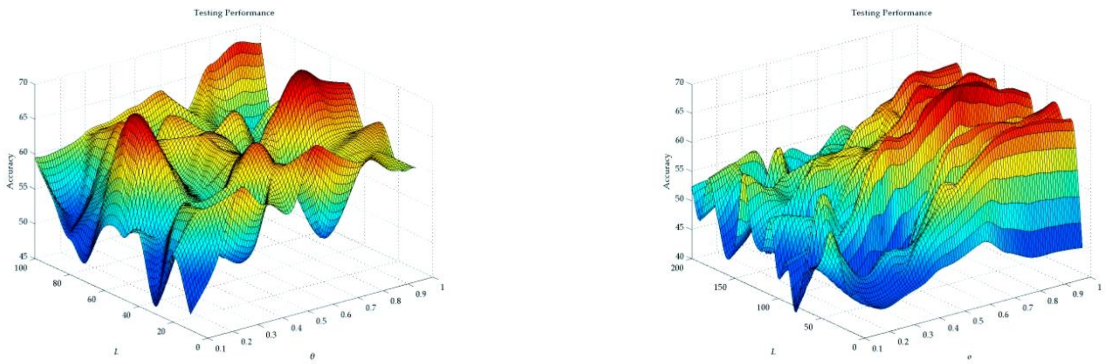
(a) On Auto Mpg data set



(b) On Ecoli data set



(c) On Parkinsons data set



(d) On Sonar data set

Fig. 4. Impact of parameter pairs (L, θ) and (L, σ) on **testing** performances of LCELM and improved LCELM

6. Conclusion

This paper presented an improved LCELM algorithm which overcomes two shortcomings of the existing LCELM by assigning the nonrandom address for hidden-layer node and calculating the activated degree with Gaussian kernel. The experimental results confirmed that the improved LCELM had the better generalization performances than ELM and LCELM. In our future works, we will apply the improved LCELM to handle the fuzzy nonlinear regression analysis [14] [15] and uncertainty mining from big data [16] [17].

Acknowledgment

We thank Editor and anonymous reviewers whose valuable comments and suggestions help us to improve this paper significantly. The corresponding author, Dr. Yu Lin He, is supported by China Postdoctoral Science Foundation (2015M572361) and National Natural Science Foundation of China (61503252).

References

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

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

[3] Fu, A. M., Dong, C. R., & Wang, L. S. (2015). An experimental study on stability and generalization of extreme learning machines. *International Journal of Machine Learning and Cybernetics*, *6(1)*, 129-135.

[4] Huang, G. B., Song, S. J., Gupta, J. N., & Wu, C. (2014). Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics*, *44(12)*, 2405-2417.

[5] Zong, W., & Huang, G. B. (2011). Face recognition based on extreme learning machine. *Neurocomputing*, *74(16)*, 2541-2551.

[6] Liu, H., Yu, L., Wang, W., & Sun, F. (2016). Extreme learning machine for time sequence classification. *Neurocomputing*, *174*, 322-330.

[7] Tang, J. X., Deng, C. W., & Huang, G. B. (2016). Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4), 809-821.

[8] Huang, G. B., Wang, D. H., & Lan, S. Y. (2011). Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics*, *2(2)*, 107-122.

[9] Huang, G., Huang, G. B., Song, S., & You, K. (2015). Trends in extreme learning machines: A review. *Neural Networks*, *61*, 32-48.

[10] Qu, Y. (2016). Local coupled extreme learning machine. *Neural Computing and Applications*, *27(1)*, 27-33.

[11] Sun, J. (2010). Local coupled feedforward neural network. *Neural networks*, *23(1)*, 108-113.

[12] UCI machine learning repository. Retrieved from http://archive.ics.uci.edu/ml/

[13] Solving tan(x)=x. Retrieved from http://blitiri.blogspot.com/2012/11/solving-tan-x-x.html

[14] He, Y. L., Wang, X. Z., & Huang, J. Z. X. (2016). Fuzzy nonlinear regression analysis using a random weight network. *Information Sciences*.

[15] Liu, H. T., Wang, J., He, Y. L., & Ashfaq, R. A. R. (2016). Extreme learning machine with fuzzy input and fuzzy output for fuzzy regression. *Neural Computing and Applications*.

[16] Wang, X. Z. (2015). Uncertainty in learning from big data-editorial. *Journal of Intelligent & Fuzzy Systems*, *28(5)*, 2329-2330.

[17] Wang, X., & Huang, J. Z. (2015). Editorial: Uncertainty in learning from big data. *Fuzzy Sets and Systems*,

258, 1-4.

Chong Liu received his bachelor degree from Tianjin University of Science & Technology in June 2005. Since July 2005, he has been a lecturer of Cangzhou Technical College (CTC), Cangzhou, Hebei, China. His research interests include machine learning and pattern recognition.

Bing Qiang Wang received his bachelor degree from Agricultural University of Hebei Province in June 2004 and received his master degree from Yanshan University in June 2013. Since July 2004, he has been a lecturer of Cangzhou Technical College (CTC), Cangzhou, Hebei, China. His research interests include machine learning and pattern recognition.

Xiao Lan Wang received her master degree from Hebei University in June 2009. Since September 2011, she has been a lecturer of Cangzhou Technical College (CTC), Cangzhou, Hebei, China. Her research interests include machine learning and pattern recognition.

Yu Lin He received both his master degree in computer science and a Ph.D. degree in optical engineering from Hebei University in June 2009 and June 2014, respectively. He is currently a post-doctoral fellow in College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, China. From February 2011 to January 2012 and from August 2013 to August 2014, he has served as a research assistant in the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include random weights networks, extreme learning machine, artificial neural networks, evolutionary optimization, probability density function estimation, and Bayesian network.

Rana Aamir Raza Ashfaq received his master degree in computer science from Blekinge Tekniska Hgskola (BTH), Sweden in 2009. He also received his bachelor and a master degrees in computer science from Bahauddin Zakariya University, Multan, Pakistan in 2000 and 2003 respectively. Since 2010, he is working as the assistant professor in Department of Computer Science, Bahauddin Zakariya University. He is currently a Ph.D. student in College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guangdong, China. His main research interests include machine learning and big data mining.