Neural Comput & Applic
DOI 10.1007/s00521-013-1542-4

Local coupled extreme learning machine

Yanpeng Qu

Received: 11 September 2013 / Accepted: 24 December 2013

Abstract Due to the significant efficiency and simple implementation, extreme learning machine (ELM) algorithms enjoy much attention in regression and classification applications recently. Many efforts have been paid to enhance the performance of ELM from both methodology (ELM training strategies) and structure (incremental or pruned ELMs) perspectives. In this paper, a local coupled extreme learning machine (LC-ELM) algorithm is presented. By assigning an address to each hidden node in the input space, LC-ELM introduces a decoupler framework to ELM in order to reduce the complexity of the weight searching space. The activated degree of a hidden node is measured by the membership degree of the similarity between the associated address and the given input. Experimental results confirm that the proposed approach works effectively and generally outperforms the original ELM in both regression and classification applications.

Keywords Extreme learning machine · LC-ELM · Classification · Regression

1 Introduction

Extreme learning machine (ELM) [1, 2] algorithms have recently enjoyed much attention as a powerful tool in performing regression and classification tasks (e.g. [3, 4]). A variety of the ELM-based methods, therefore, have been developed in attempt to improve the performance. Generally, there are two ways: one is to optimise the methodology of ELM (e.g. online sequential ELM [5], evolutionary ELM [6], partial Lanczos ELM [7]); the other is to refine the distribution of the hidden nodes in ELM in order to result in a more reasonable learning model (e.g. incremental ELM [8], pruned ELM [9], optimally pruned ELM [10], two-stage ELM [11]). Overall, at the theoretical and empirical levels, both of these two schemes have achieved remarkable performance.

The local coupled feedforward neural network (LCFNN) algorithm was introduced in [12]. In LCFNN, each hidden node is assigned an address in the input space. The distance induced by such address to the input is considered to locate the position of corresponding hidden nodes in the area of a window function. Since this input will only activate the hidden nodes within the radium of a window function, the fully connected structures of multilayer perceptrons get decoupled. This method ensures that the dimensionality of the weight searching space gets limited and the corresponding convergence rates of the training algorithms (e.g. BP [13], GA [14]) are successfully accelerated. As the further developments of LCFNN, in [15], the learning efficiency of the LCFNN binary classifier was enhanced, and a modified gradient-based learning algorithm was proposed to further improve the performance of LCFNN in [16].

This paper presents a novel approach entitled local coupled extreme learning machine (LC-ELM). Such algorithm ulteriorly develops the LCFNN algorithm in structure and employs the ELM algorithm to implement the mechanism. Specifically, in LC-ELM, the similarity degrees between the hidden nodes and the given inputs are measured by a similarity relation (e.g. fuzzy similarity [17], kernel function [18]), and the activated degree of each hidden node is determined by a fuzzy membership function. In so doing, the fully coupled linking architectures of ELM is softly simplified rather than brutally segmented,

Y. Qu (✉)
Information Science and Technology College,
Dalian Maritime University, Dalian 116026, China
e-mail: yanpengqu@dlmu.edu.cn

Published online: 23 January 2014

and the revision of the inputs will influence the different components of ELM with varying degrees. Consequently, the quantity and quality of the activated hidden nodes in LC-ELM can be refined by the preset fuzzy membership function. It is noteworthy that the existence of a great number of fuzzy membership functions and the means of similarity measurements ensure the LC-ELM method is highly flexible in dealing with different problems. Empirical evaluations demonstrate that with such efficient mechanism, LC-ELM outperforms the classical ELM methods in both regression and classification tasks.

The reminder of this paper is structured as follows. An outline of the relevant background material is presented in Sect. 2, including the classical ELM and the local coupled feedforward neural networks techniques. The local coupled extreme learning machine is then described in Sect. 3 and systematically compared against the ELM methods in an experimental evaluation in Sect. 4. Section 5 concludes the paper with a short discussion of the potential further works.

2 Theoretical background

2.1 Extreme learning machine

The ELM algorithm was originally proposed in [19]. It is implemented by employing a single-hidden layer feedforward neural network (SLFN) structure. The underlying mechanism of the ELM lies in the random initialization of the SLFN weights and biases. Then, without tuning the input weights and biases, the output weights are determined by ordinary least-square methods. The network is thereby obtained with very few steps and very low computational cost.

For a dataset which contains M distinct objects: $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $\mathbf{t}_i \in \mathbb{R}^q$, an N-hidden-node nonlinear SLFN system can be modelled as the following sum:

$$\sum_{i=1}^{N} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i), \quad j = 1, \ldots, M, \tag{1}$$

where $g(\cdot)$ denotes the activation function, $\mathbf{w}_i$ and b_i are the parameters of hidden layer and $\boldsymbol{\beta}_i$ is the output weights.

In the case where SLFNs can approximate these M samples with zero error and the corresponding relation is

$$\sum_{i=1}^{N} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \ldots, M, \tag{2}$$

the above M equations can be written compactly as a linear system:

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

where $\mathbf{H} = \{h_{ij}\}$ $(i = 1, \ldots, M, j = 1, \ldots, N)$ is the hidden layer output matrix, $h_{ij} = g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j)$ denotes the output of the jth hidden neuron with respect to $\mathbf{x}_i$. $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N]^T_{N\times q}$ is the matrix of output weights and $\boldsymbol{\beta}_j$ denotes the weight vector connecting the jth hidden node and the output layer. $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_M]^T_{M\times q}$ is the matrix of target outputs.

Thus, training a SLFN is equivalent to finding the minimal Euclidean norm solution $\boldsymbol{\beta}$ of the linear system (3):

$$\widehat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}, \text{ such that } \boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2 \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalised inverse [20, 21] of the hidden layer output matrix $\mathbf{H}$.

Following the above discussion, a three-step ELM algorithm can be summarised in Fig. 1.

2.2 Local coupled feedforward neural networks

The general structures of local coupled feedforward neural networks are similar to those of multilayer perceptrons (MLP). The differences between them rest in their working and the associated learning methods. For MLP, each input activates all hidden nodes with an equal intensity. This generates a fully coupled structure that leads to the computation cost in proportion with the scale of a given network. To overcome this shortcoming of MLP, in LCFNN, each hidden node is assigned an address within an input space, and each activates hidden nodes through an intensity function $G(\|\mathbf{x} - \mathbf{d}_i\|)$, where $\|\mathbf{x} - \mathbf{d}_i\|$ is the distance between the input and the ith hidden node (with the address $\mathbf{d}_i$), and $G(\cdot)$ is a window function which satisfies the following conditions [12]:

1. $G(\cdot)$ is a continuous function,
2. $G(0) = 1$,
3. $G(\cdot) = 0$ in $[r, +\infty)$
4. $G(\cdot)$ is a monotonic function in $[0,r)$, where $r > 0$) is the window radius.

With p input nodes $\mathbf{x} = (x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$, N hidden nodes and q output nodes, the output of a single-hidden layer LCFNN is:

ELM($\mathbb{N}$,g,N)
$\mathbb{N}$, the training set $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^p, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \cdots, M\}$,
g, the activation function,
N, the number of hidden nodes.

(1) Randomly assign hidden node parameters $(\mathbf{w}_j, b_j)$, $j = 1, \cdots, N$,
(2) Calculate the hidden layer output matrix $\mathbf{H}$,
(3) Calculate the output weight $\boldsymbol{\beta}$.

Fig. 1 The ELM algorithm

$$\sum_{i=1}^{N}(\boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x} + b_i) + c_i)G(\|\mathbf{x} - \mathbf{d}_i\|), \tag{5}$$

where $\boldsymbol{\beta}_i$, b_i, c_i and $\mathbf{w}_i$ are the network parameters, $g(\cdot)$ is the activation function.

In LCFNN, given the definition of the window function, as the distance between a hidden node and the input grows, the activated degree of this hidden node is monotonically decreasing. In particular, when this distance is greater than the radius r of the window function, the relevant activated intensity of this hidden node will become zero. Such observation corresponds to removing this hidden node from the LCFNN. In so doing, the upper bound of the number of the hidden nodes of LCFNN can be controlled effectively.

3 Local coupled extreme learning machine

A modified gradient-based learning method was proposed to improve the performance of LCFNN [16]. In order to ulteriorly develop the LCFNN algorithm in structure, a novel algorithm entitled LC-ELM is presented in this work.

3.1 Properties

The linking architecture of LC-ELM is similar to that of LCFNN. Rather than the hard window function $G(\cdot)$ and the norm distance $\|\mathbf{x} - \mathbf{d}_i\|$ in LCFNN, the fuzzy membership function $F(\cdot)$ and the similarity relation $S(\mathbf{x}, \mathbf{d}_i)$ are employed to implement LC-ELM.

Again, for a dataset that contains M distinct objects: $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i \in \mathbb{R}^p$ and $\mathbf{t}_i \in \mathbb{R}^q$, the output of an N-hidden-node nonlinear LC-ELM is:

$$\sum_{i=1}^{N}\boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i)F(S(\mathbf{x}_j, \mathbf{d}_i)), \quad j = 1, \ldots, M, \tag{6}$$

where $g(\cdot)$ denotes the activation function, $\mathbf{w}_i$, b_i and $\boldsymbol{\beta}_i$ are network weights, $\mathbf{d}_i$ is the address of the ith hidden nodes.

The similarity relation $S(\mathbf{x}, \mathbf{d}_i)$ enjoys a wide range of choice. In fact, it may cover various types of binary relation. In addition to the p-norm distance $(p = 1, 2, +\infty)$ [15], fuzzy similarity relations [17], fuzzy tolerance relations [22], kernel functions [18], etc., are all alternatives to implement LC-ELM. From a statistics perspective, generally, two important classes of kernels are stationary kernels and nonstationary kernels [23]. In particular, isotropic stationary kernels depend only on the norm of the lag vector (i.e. $\|\mathbf{x} - \mathbf{d}_i\|$). In this case, isotropic stationary kernels basically share the same measurement terms with p-norm distance.

In LC-ELM, the fuzzy membership function $F(\cdot)$ is defined with following properties:

1. $F(\cdot)$ is a nonnegative piecewise continuous function,
2. $F(\cdot)$ is monotonically decreasing in $[0, +\infty)$,
3. $F(0) = 1$,
4. $F(x) \to 0$, $x \to +\infty$.

Here, $F(\cdot)$ 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 [2]. In order to adjust the width of the activation area, the underlying radius parameter r is remained in $F(\cdot)$ as well. In this case, the $G(\cdot)$ in LCFNN can be considered as a special case of $F(\cdot)$, when $F(x) = 0$, $x > r$.

Different from LCFNN, in LC-ELM, the hidden nodes which locate out of the sphere, whose centre is the input point and the radius is r, will be still weakly activated rather than brutally omitted. Because the new samples (e.g. testing data) only highly activate the trained hidden nodes located inside the radius r, the rest weakly inspired hidden nodes will not contribute too much for the system decision any longer. This decoupler process is similar to the learning process of a brain: when a new learning sample is achieved, only relative knowledge need be revised with different memory inspired degrees.

It is noteworthy that in (6), when the $F(\cdot)$ is a constant function which is equal to 1, LC-ELM is reduced to the classical ELM. Moreover, when $\mathbf{w}_i$ in (6) is equal to zero, the fuzzy membership function $F(\cdot)$ is nonconstant, and the similarity function $S(\mathbf{x}, \mathbf{d}_i)$ is determined by the norm distance $\|\mathbf{x} - \mathbf{d}_i\|$, and then, the framework of LC-ELM is reduced to the ELM with RBF hidden nodes [2]. In [8], both of these two cases of ELM are proven to own universal approximation capabilities. Therefore, it is reasonable to consider that for an arbitrary multivariate continuous function, LC-ELMs may have the ability to approximate the function under a given accuracy.

3.2 Algorithmic implementation

Given the presentation in the Eq. (6), generally, LC-ELM shares a similar three-step methodology with the classical ELM (as shown in Fig. 1). In the initialisation phase of LC-ELM, besides the parameters $(\mathbf{w}_i, b_i)$ of the hidden layer, the address $\mathbf{d}_i$ of each hidden node is assigned randomly as well.

In so doing, for the linear system generated by LC-ELM,

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{7}$$

the hidden layer output matrix in LC-ELM becomes

$$\mathbf{H} = \left\{h_{ij} = g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i)F(S(\mathbf{x}, \mathbf{d}_i))\right\} \quad i = 1, \ldots, M, \; j = 1, \ldots, N. \tag{8}$$

A collection of certain commonly used similarity measurements [23, 24] is listed as follows:

- p-norms: $\|\mathbf{x}-\mathbf{y}\|_p, \quad (p=1,2,+\infty)$
- Kernel functions:
  - Gaussian kernel $S(\mathbf{x},\mathbf{y})=\exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\theta}\right)$
  - Exponential kernel $S(\mathbf{x},\mathbf{y})=\exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|}{\theta}\right)$
  - Rational quadratic kernel $S(\mathbf{x},\mathbf{y})=1-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\|\mathbf{x}-\mathbf{y}\|^2+\theta}$
  - Wave kernel $S(\mathbf{x},\mathbf{y})=\frac{\theta}{\|\mathbf{x}-\mathbf{y}\|}\sin(\frac{\|\mathbf{x}-\mathbf{y}\|}{\theta})$,
  - Polynomial kernel: $S(\mathbf{x},\mathbf{y})=(a\mathbf{x}\cdot\mathbf{y}+c)^d$,here, Gaussian kernel, exponential kernel, rational quadratic kernel and wave kernel are isotropic stationary kernels [23]. Polynomial kernel is one of the typical nonstationary kernel functions.
- *Fuzzy similarity* $S(\mathbf{x},\mathbf{y})=T_{a\in P}\{\mu_{R_a}(\mathbf{x},\mathbf{y})\}$,where T is a T-norm. P is a subset of features, $\mu_{R\_a}(\mathbf{x},\mathbf{y})$ is the degree to which objects $\mathbf{x}$ and $\mathbf{y}$ are similar for feature a.

As well as the similarity relations, the fuzzy membership functions in LC-ELM enjoy a variety of forms. For instance, triangular function (9), z-shape membership function (10) [25], gaussian function (11), the reversed sigmoid function (12) and reversed tanh function (13) are alternative choices in practice.

$$F(x)=\begin{cases}\frac{r-x}{r} & r\geq x\geq 0\\ 0 & x>r\end{cases}, \tag{9}$$

$$F(x)=\begin{cases}1 & x\leq a\\ 1-2\frac{(x-a)^2}{(a-b)^2} & a<x\leq\frac{a+b}{2}\\ 2\frac{(b-x)^2}{(a-b)^2} & \frac{a+b}{2}<x\leq b\\ 0 & x>b\end{cases}, \tag{10}$$

$$F(x)=\exp\left(-\frac{x^2}{r}\right), \tag{11}$$

$$F(x)=\frac{2}{1+\exp(\frac{x}{r})}, \tag{12}$$

$$F(x)=\tanh(-\frac{x}{r})+1. \tag{13}$$

4 Experimental evaluation

This section presents a systematic evaluation of LC-ELM experimentally. The results and discussions are divided into three different parts. The first part compares LC-ELM with the classical ELM in the aspects of function approximation. The comparison between LC-ELM and the ELM with RBF hidden nodes on real-world regression problems is carried out in the second part. The third part provides a comparative investigation of the classification performance of LC-ELM against ELM on several benchmark datasets. Note that, the fuzzy membership functions and the similarity relations used to implement the following LC-ELM methods are assigned empirically.

4.1 Function regression

The task is to approximate the nonlinear function

$$f(x_1,x_2)=\cos\left(0.6\sqrt{x_1^2+x_2^2}\right). \tag{14}$$

In this example, 51 × 51 training and testing patterns are stochastically selected from a $[-10, 10]\times[-10, 10]$ square region, respectively. The parameters of the hidden layers in both LC-ELM and ELM are all randomly chosen in $[-1, 1]$. The addresses of the hidden nodes in LC-ELM are assigned as the random value in [0, 1]. The sigmoid function $g(x)=1/(1+\exp(-x))$ is utilised as the activation function in the hidden layers of both LC-ELM and ELM. The fuzzy membership in LC-ELM is set to be reversed sigmoid function (12) with $r=0.4$. And the wave kernel is employed to measure the similarity between the input and the hidden nodes in LC-ELM.

A series of experiments are carried out in order to ascertain the variation in the resulting regression root-mean-squared error (RMSE) by changing the number of the hidden nodes in LC-ELM and ELM, with N being the tens ranging from 10 to 300. For each N, 10 trials have been conducted for both LC-ELM and ELM, and the average testing RMSEs are illustrated in Fig. 2. Furthermore, across all the 30 numbers of the hidden nodes in Fig. 2, the lowest results (denoted by the red markers), the associated numbers of hidden nodes and the average training and testing results with the standard deviations (SD) of LC-ELM and ELM are summarised in Table 1, respectively.

It can be observed from the Fig. 2 and Table 1 that LC-ELM outperforms ELM at the levels of both the lowest and the average RMSEs. In particular, LC-ELM reaches the better result than ELM with less hidden nodes. This indicates that compared with the classical ELM, LC-ELM enjoys a faster training model structurally, due to its simpler weight searching space.

4.2 Real-world regression problems

This section presents the results of a comparison of LC-ELM with ELM on the benchmark regression datasets taken from UCI machine learning repository [26] and Statlib [27]. The specification of the tested datasets is shown in Table 2.

In this experiment, 10 trials are conducted for each problem. The training and testing data of the corresponding datasets are reshuffled at each trial of simulation. The

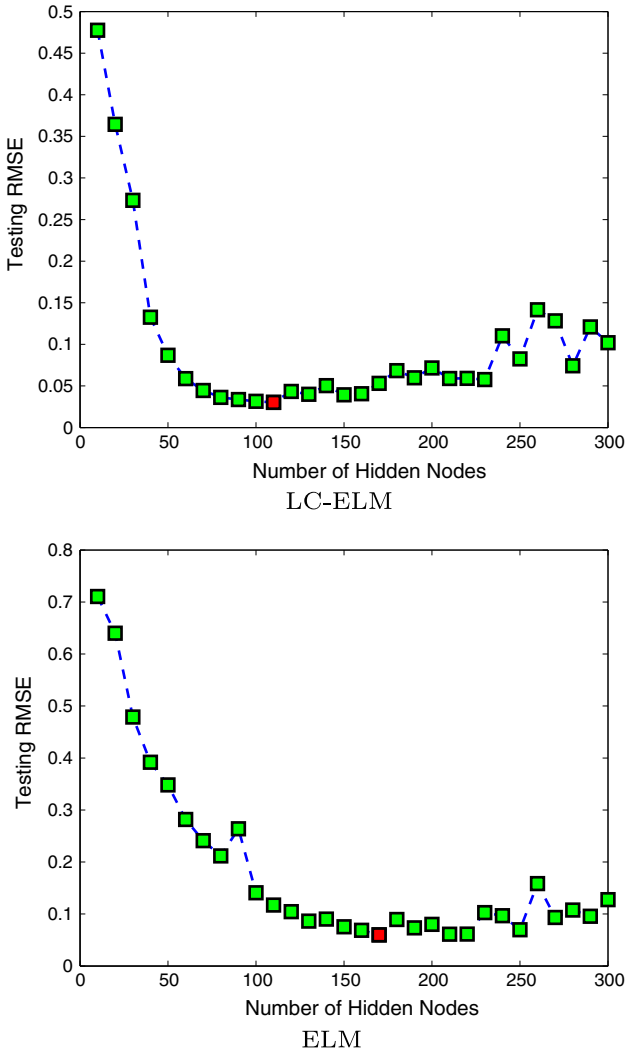

Fig. 2 Testing RMSEs of LC-ELM and ELM with respect to different numbers of hidden nodes

Table 1 Regression results of LC-ELM and ELM

Algorithm	Lowest testing RMSE	Hidden nodes	Avg. training		Avg. testing	
			RMSE	SD	RMSE	SD
LC-ELM	0.0302	110	0.1390	0.0083	0.0991	0.0391
ELM	0.0599	170	0.1847	0.0139	0.1842	0.0619

configurations of LC-ELM and ELM are roughly the same as those in the Sect. 4.1, except the window function of LC-ELM here applies reversed tanh function (13), and the ELM algorithms are constructed with RBF hidden nodes with multiquadric function $g(x) = (\|x - a\|^2 + b^2)^{1/2}$. The average RMSE and SD over the training data and the testing data across the 10 trials are listed in Table 3. For each dataset in Table 3, the same number of hidden nodes is applied in LC-ELM and ELM.

Table 2 Specification of tested regression problems

Datasets	# Attributes	# Training data	# Testing data
Abalone	8	2,784	1,393
Ailerons	5	4,752	2,377
Autoprice	9	106	53
Bodyfat	14	168	84
Cloud	9	682	342
Computer	12	5461	2,731
CPU	6	139	70
Elevators	6	6344	3173
Housing	13	377	169
Stock	9	633	317

In Table 3, the superior RMSE results, which are lower than their counterparts by more than 0.005, will be shown in boldface. It can be seen from these results that, compared to ELM, LC-ELM generally performs better with lower RMSE and SD results. Occasionally, on *Ailerons*, *Computer*, *Elevators* and *Stock* datasets, the training RMSE achieved by LC-ELM is slightly greater than those of ELM. However, in the relevant testing comparisons, LC-ELM outperforms ELM on all tested datasets. In particular, on *Autoprice* and *CPU* datasets, LC-ELM gains significant improvements compared with ELM.

4.3 Classification problems

In this section, the performances of the LC-ELM method will be compared against those of the ELM algorithm on several benchmark classification datasets [26]. The specification of the datasets is displayed in Table 4.

Again, in this experiment, each problem will run 10 trails with reshuffling the training and testing data. Different from the configuration in the Sect. 4.2, the sigmoid hidden nodes will be adopted in ELM and the window radius in LC-ELM is assigned to be 0.7 in this example. Again, the numbers of hidden node used in both LC-ELM and ELM, the average classification accuracy (Accy) and SD over the training data and the testing data across the 10 trials are listed in Table 5. The same numbers of hidden nodes are used in LC-ELM and ELM.

In Table 5, the superior classification accuracy, which are higher than their counterparts by more than 0.005, will be shown in boldface as well. Overall, the LC-ELM method outperforms the ELM algorithm. Compared with ELM, LC-ELM yields the better performance on 9 of the 10 datasets. Occasionally, for *Iris*, LC-ELM results in a lower classification accuracy than ELM.

In summary, examining all of the results obtained, including those for RMSE, it is clear that with a simplified weight searching space, LC-ELM offers an efficient

Table 3 Regression results of LC-ELM and ELM

Datasets	Hidden nodes	ELM training		LC-ELM training		ELM testing		LC-ELM testing	
		RMSE	SD	RMSE	SD	RMSE	SD	RMSE	SD
Abalone	25	0.0761	0.0012	0.0754	0.0011	0.0776	0.0027	0.0767	0.0018
Ailerons	45	0.0380	0.00042	0.0382	0.00042	0.0394	0.00083	0.0386	0.00065
Autoprice	15	0.0879	0.0101	**0.0757**	0.0044	0.1205	0.0190	**0.0842**	0.0087
Bodyfat	50	0.0678	0.0038	0.0661	0.0038	0.1295	0.0191	**0.1243**	0.0243
Cloud	25	0.0133	0.0013	0.0116	0.00086	0.0179	0.0033	0.0135	0.0022
Computer	125	0.0339	0.00079	0.0345	0.0016	0.0408	0.0044	0.0407	0.0033
CPU	10	0.0476	0.0066	**0.0416**	0.0070	0.0865	0.0584	**0.0582**	0.0203
Elevators	125	0.0518	0.00028	0.0520	0.00036	0.0536	0.00059	0.0528	0.00067
Housing	50	0.0793	0.0047	**0.0711**	0.0035	0.0929	0.0093	0.0884	0.0091
Stock	110	0.0261	0.0010	0.0269	0.0011	0.0332	0.0021	0.0321	0.0012

Table 4 Specification of tested classification problems

Datasets	# Attributes	# Training data	# Testing data	# Class
Escherichia coli	7	224	112	8
Glass	9	142	72	7
Heart	13	180	90	2
Ionosphere	34	153	77	2
Iris	4	100	50	3
Liver	6	230	115	2
Olitos	25	80	40	4
Sonar	60	138	70	2
Wine	14	118	60	3
Wisconsin	9	455	228	2

learning strategy to solve the regression and classification problem. Additionally, given the large number of the similarity relations and the fuzzy membership functions, LC-ELM can be implemented into various forms, e.g. the classical ELMs with sigmoid and RBF hidden nodes. This mechanism allows LC-ELM has the ability to flexibly generate a solution for different problems.

5 Conclusions

In order to reduce the complexity of the weight searching space in ELM, this paper presents a LC-ELM algorithm. In LC-ELM, the fuzzy membership functions and the similarity relations are employed to measure the activated degree of each node in the hidden layer. With such degree, the hidden nodes that are located out of the activation area of the inputs will be reduced or even omitted. And this scenario successfully limits the dimensionality of weight searching space in forming the algorithmic model and leads a efficient and effective learning approach.

In addition, due to the massive existence of the fuzzy membership functions and the similarity relations, LC-

Table 5 Classification results of LC-ELM and ELM

Datasets	Hidden nodes	ELM training (%)		LC-ELM training (%)		ELM testing (%)		LC-ELM testing (%)	
		Accy	SD	Accy	SD	Accy	SD	Accy	SD
E. coli	20	89.78	0.90	89.29	1.17	85.71	1.68	**87.14**	4.23
Glass	20	72.25	3.82	**74.79**	2.97	63.47	5.36	63.75	6.56
Heart	20	86.61	2.52	86.83	1.46	82.44	3.70	**83.22**	2.43
Ionosphere	40	91.70	1.69	**93.86**	2.05	83.51	3.12	**86.49**	2.68
Iris	15	98.30	0.67	97.90	1.29	96.60	2.50	96.40	2.07
Liver	30	**77.78**	1.22	76.43	2.49	70.17	2.53	**72.52**	4.04
Olitos	20	94.25	2.37	94.50	2.14	78.25	5.78	**79.75**	7.50
Sonar	40	88.70	2.92	**89.42**	4.06	75.57	3.95	**76.29**	6.25
Wine	20	99.92	0.27	99.83	0.36	98.33	1.36	**98.83**	1.37
Wisconsin	40	97.71	0.55	97.49	0.43	96.36	1.42	**96.97**	1.06

ELM enjoys a variety of implementations. For instance, both classical ELM and RBF-ELM can be implemented by LC-ELM with certain configurations. This fact ensures the flexibility and positivity of LC-ELM when solving different problems.

Topics for further research include a more comprehensive study of how LC-ELM would perform with other fuzzy membership functions and similarity relations [28] as the alternative. Correspondingly, the sensitivity of these chosen functions is also necessary to be exploited in theory. In addition, the addresses of the hidden nodes in LC-ELM are randomly chosen in this paper. In order to refine the framework of LC-ELM reasonably, an adaptive algorithm for assigning those addresses remains an active research in both of the theoretical and practical views. Actually, in [15], an optimisation scheme of the hidden nodes and the associated address were discussed for LCFNN. It is worth trying to introduce such method to LC-ELM in the future investigations. Furthermore, a more complete comparison of LC-ELM against the traditional ELM algorithm and other techniques over different datasets from real application domains [4, 29] would form the basis for a wider series of topics for future studies.

Acknowledgments This work is jointly supported by the National Natural Science Foundation of China (61272171), the Fundamental Research Funds for the Central Universities (No. 3132013335) and the China Postdoctoral Science Foundation (2013M541213).

References

1. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
2. Huang G-B, Wang D, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2:107–122
3. Yang Y, Wang Y, Yuan X (2012) Bidirectional extreme learning machine for regression problem and its learning effectiveness. IEEE Trans Neural Netw Learn Syst 23:1498–1505
4. Qu Y, Shang C, Wu W, Shen Q (2011) Evolutionary fuzzy extreme learning machine for mammographic risk analysis. Int J Fuzzy Syst 13:282–291
5. Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17:1411–1423
6. Zhu Q-Y, Qin AK, Suganthan PN, Huang G-B (2005) Evolutionary extreme learning machine. Pattern Recognit 38:1759–1763
7. Tang X, Han M (2009) Partial lanczos extreme learning machine for single-output regression problems. Neurocomputing 72:3066–3076
8. Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17:879–892
9. Rong H-J, Ong Y-S, Tan A-H, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72:359–366
10. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: pptimally pruned extreme learning machine. IEEE Trans Neural Netw 21:158–162
11. Lan Y, Soh YC, Huang G-B (2010) Two-stage extreme learning machine for regression. Neurocomputing 73:3028–3038
12. Sun J (2010) Local coupled feedforward neural network. Neural Netw 23:108–113
13. de Jesús Rubio J, Angelov P, Pacheco J (2011) Uniformly stable backpropagation algorithm to train a feedforward neural network. IEEE Trans Neural Netw 22:356–366
14. Golmohammadi D, Creese R, Valian H, Kolassa J (2009) Supplier selection based on a neural network model using genetic algorithm. IEEE Trans Neural Netw 20:1504–1519
15. Sun J (2012) Learning algorithm and hidden node selection scheme for local coupled feedforward neural network classifier. Neurocomputing 79:158–163
16. Qu Y, Shang C, Yang J, Wu W, Shen Q (2013) Modified gradient-based learning for local coupled feedforward neural networks with gaussian basis function. Neural Comput Appl 22:379–394
17. Radzikowska A, Kerre E (2002) A comparative study of fuzzy rough sets. Fuzzy Sets Syst 126:137–155
18. Shawe-Taylor J, Christianini N (2004) Kernel method for pattern analysis. Cambridge University Press, Cambridge
19. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the international joint conference on neural networks (IJCNN2004), vol 2, Budapest, Hungary, pp 985–990
20. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
21. Serre D (2002) Matrices: theory and applications. Springer, Berlin
22. Das M, Chakraborty M, Ghoshal T (1998) Fuzzy tolerance relation, fuzzy tolerance space and basis. Fuzzy Sets Syst 97:361–369
23. Genton M (2001) Classes of kernels for machine learning: a statistics perspective. J Mach Learn Res 2:299–312
24. Jensen R, Shen Q (2009) New approaches to fuzzy-rough feature selection. IEEE Trans Fuzzy Syst 17:824–838
25. Chen L, Chen CLP, Pedrycz W (2010) A gradient-descent-based approach for transparent linguistic interface generation in fuzzy models. IEEE Trans Syst Man Cybern Part B Cybern 40:1219–1230
26. Blake C, Merz C (1998) UCI repository of machine learning databases. University of California, Irvine
27. Mike M (1989) Statistical datasets. Carnegie Mellon University, Pittsburg
28. Qu Y, Shang C, Shen Q, Mac Parthálain N, Wu W (2011) Kernel-based fuzzy-rough nearest neighbour classification. In: IEEE international conference on fuzzy systems (FUZZ), pp 1523–1529
29. Shang C, Barnes D, Shen Q (2011) Facilitating efficient mars terrain image classification with fuzzy-rough feature selection. Int J Hybrid Intell Syst 8:3–13