Cogn Comput (2016) 8:720–728
DOI 10.1007/s12559-016-9409-5

Self-adaptive Extreme Learning Machine Optimized by Rough Set Theory and Affinity Propagation Clustering

Li Xu[1,2] · Shifei Ding[1,2] · Xinzheng Xu[1,2] · Nan Zhang[1,2]

Received: 30 October 2014 / Accepted: 8 April 2016 / Published online: 18 April 2016

Abstract Recently, a simple and efficient learning algorithm for single hidden layer feedforward networks (SLFNs) called extreme learning machine (ELM) has been developed by G.-B. Huang et al. One key strength of ELM algorithm is that there is only one parameter, the number of hidden nodes, to be determined while it has the significantly low computational time required for training new classifiers and good generalization performance. However, there is no effective method for finding the proper and universal number of hidden nodes. In order to address this problem, we propose a self-adaptive extreme learning machine (SELM) algorithm. SELM algorithm determines self-adaptively the number of hidden nodes and constructs Gaussian function as activation functions of hidden nodes. And in this algorithm, rough set theory acts as the pretreatment cell to eliminate the redundant attributes of data sets. Then, affinity propagation clustering (AP Clustering) is used to self-adaptively determine the number of hidden nodes, while the centers and widths of AP clustering are utilized to construct a Gaussian function in the hidden layer of SLFNs. Empirical study of SELM algorithm on several commonly used classification benchmark problems shows that the proposed algorithm can find the proper number of hidden nodes and construct compact network classifiers, comparing with traditional ELM algorithm.

✉ Shifei Ding
dingsf@cumt.edu.cn

[1] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

[2] Jiangsu Key Laboratory of Mine Mechanical and Electrical Equipment, China University of Mining & Technology, Xuzhou 221116, China

Keywords Extreme learning machine · Self-adaptive extreme learning machine · Rough set theory · AP clustering

Introduction

Feedforward neural networks have been extensively studied and used in many applications for their learning capabilities and generalization ability, such as classification [1], regression [2], pattern recognition [3], machine learning [4], control [5] and fault diagnosis [6]. However, the conventional learning algorithms for SLFNs have some inherent drawbacks: lower learning efficiency, being easy to lose in the local minimum and overfitting. In order to overcome these drawbacks, Huang et al. [7] proposed a novel learning algorithm called extreme learning machine (ELM) for single hidden layer feedforward networks (SLFNs). In ELM algorithm, the weights linked the input layer and the hidden layer and the thresholds of hidden neural nodes are unnecessarily tuned iteratively in the training process but randomly generated at the beginning of learning [8]. The unique optimal solution can be obtained as long as the number of hidden neurons is determined [9]. Recently, many researchers regard ELM as a learning method for multi-class classification, regression and pattern clustering [10–12]. ML-ELM proposed by Kasun et al. [13] stacks extreme learning machine-based autoencoder (ELM-AE) to create a multilayer neural network. Bai et al. [14] presented sparse ELM which can reduce storage space and testing time significantly. Huang et al. [15] also introduced manifold regularization framework into the ELM model and proposed semi-supervised ELM (SS-ELM) and unsupervised ELM (US-ELM). And Lin [16] presented the theoretical analysis of ELM. Compared with other

conventional training algorithms for neural networks, ELM algorithm has lots of advantages, such as fast learning speed and good generalization performance.

However, in the standard ELM algorithm, the number of hidden neurons must be determined in advance even though there is no priori knowledge about the classification problem being solved. To overcome above problems, the pruned-ELM (P-ELM) and the incremental ELM (I-ELM) algorithms were proposed as the typical methods. In view of too few/many hidden nodes employed that would lead to underfitting/overfitting issues in pattern classification, Rong et al. [17] presented a pruned-ELM (P-ELM) algorithm as a systematic and automated approach for designing ELM network. P-ELM began with a large initial number of hidden nodes and then removed the irrelevant or lowly relevant hidden nodes by considering their relevance to the class labels during learning. As a result, the architectural design of ELM can be automated. On the other hand, Huang et al. [18] proposed an I-ELM algorithm to construct an incremental feedforward network. I-ELM randomly added the hidden layer nodes one by one and froze the output weights of the existing hidden nodes when a new hidden node was added. I-ELM is not only efficient for SLFN with continuous activation functions (including differentiable), but also efficient for SLFNs with piecewise continuous activation functions (such as the threshold). In addition, on the basis of I-ELM, the convex I-ELM (CI-ELM) [19] and enhance I-ELM (EI-ELM) [20] algorithms were presented. To sum up, P-ELM and I-ELM algorithms need to adjust continuously the number of hidden neurons until they find the suitable structure of ELM. However, the adjustment procedure consumed more time. Moreover, when applying to different classification problems, the number of hidden neurons had to adjust again. So, how to find a method to self-adaptively determine the number of hidden neurons of ELM is significant and worth researching.

In this paper, we propose an improved algorithm called self-adaptive extreme learning machine (SELM) which self-adaptively determines the number of hidden neurons. In SELM, the number of hidden nodes is self-adaptively determined by the affinity propagation (AP) clustering algorithm while the centers and widths of AP clustering are utilized to construct Gaussian function, which replaces traditional activation functions (sigmoid, sine and cosine functions) of the hidden layer in ELM. SELM dramatically increases the network's learning speed, successfully avoids iteration and falling into local minimum and produces good generation performance, robustness and controllability. Experimental results indicate that SELM spends less training time and test time and improves training accuracy and testing accuracy compared with conventional ELM algorithm.

This paper is organized as follows. Section "Review of ELM" provides a brief review of the ELM. The proposed SELM algorithm is then described in section "Proposed ELM." Section "Experiments and Analyses" presents a quantitative performance comparison of SELM to other algorithms based on commonly used classification problems. Finally, the conclusions are summarized in section "Conclusions."

Review of ELM

ELM algorithm is mainly used for training the SLFNs. It sets the appropriate number of hidden nodes by continuous testing and randomly assigns the input weights and thresholds of the hidden layer. As a result, the output of the hidden layer is calculated by the activation function of hidden nodes. Then, the weights connecting the hidden layer and the output layer can be directly obtained through mathematical transformation. The learning process of ELM performs only once through this transformation and does not need any iteration. Therefore, ELM algorithm can learn significantly faster compared with the traditional back propagation algorithm (usually more than 10 times).

In the practical applications, the SLFNs needed to be trained by ELM algorithm firstly. Then, it can be used to predict the testing patterns. The SLFNs are trained by input samples including some influencing factors and their classes. When ELM algorithms are training the SLFNs, its learning process can be accomplished without any iteration. When the trained SLFNs are used to predict, it is capable of forecasting similar patterns according to its learning model.

The basic principle of ELM is outlined as follows.

A SLFN model with N arbitrary distinct samples $(x_i, t_i) \in R^n \times R^m$ $(i = 1, 2, \ldots, N)$, where $\tilde{N}$ represents the number of hidden nodes. Through activation function $f(x)$, the output of the hidden layer can be generated as follows.

$$\sum_{i=1}^{\tilde{N}} \beta_i f_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i f(a_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N \quad (1)$$

where $\alpha_i = [\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, \ldots, \alpha_{iv}]^T$ is the input weight vector linking the input layer to the ith hidden node; b_i denotes the bias of the ith hidden node; $\beta_i = [\beta_{i1}, \beta_{i2}, \beta_{i3}, \ldots, \beta_{im}]^{\mathrm{T}}$ is the output weight vector linking the ith hidden node to the output layer; and $a_i \cdot x_j$ denotes the scalar product of a_i and x_j. The activation function $f(x)$ is usually chosen from sigmoid, sine, RBF functions, etc.

Equation (1) can be succinctly written as.

$$H\beta = T \quad (2)$$

where

$$H = \begin{bmatrix} f(a_1 \cdot x_1 + b_1) & \cdots & f(a_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ f(a_1 \cdot x_N + b_1) & \cdots & f(a_{\tilde{N}} \cdot x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$$

It has been rigorously proved that SLFN is able to approximate any continuous functions with randomly assigned input weights and bias of hidden nodes while maintaining unchanged during training process, as long as the activation function $f(x)$ is infinitely differentiable in any interval and the number of hidden nodes is sufficient (generally $\tilde{N} \ll N$, in order to ensure the good generalization performance of the SLFN) [8].

After the input weights and bias of hidden nodes are randomly assigned, the output matrix of hidden layer H can be calculated by the input samples. Therefore, to train the SLFNs is equivalent to calculating the least-square (*LS*) solution of the linear equations.

$$\min_{\beta} \|H\beta - T\| \tag{3}$$

The LS solution of the linear Eq. (3) is

$$\widehat{\beta} = H^{+}T, \tag{4}$$

where H^{+} is the Moore–Penrose generalized inverse of output matrix of hidden layer H. Typically, the optimal solution $\widehat{\beta}$ contains the following features:

1. The algorithm can achieve the smallest training error;
2. The minimum paradigm of output connection weights and the optimal generalization capability of the SLFNs can be obtained;
3. The uniqueness of $\widehat{\beta}$ avoids the networks getting stuck in the local optimal solution.

In summary, the ELM algorithm for the SLFNs with training samples $(x_i,\ t_i) \in R^n \times R^m$ $(i = 1, 2, \ldots, N)$, $\tilde{N}$ hidden nodes and the activation function $f(x)$ is divided into three steps as follows.

Step 1: Determine the number $\tilde{N}$ of hidden nodes and randomly assign the input weight vector a_i and the bias b_i of hidden nodes $(i = 1, 2, \ldots, N)$.

Step 2: Calculate the output matrix $H$of the hidden layer.

Step 3: Calculate the output weights $\widehat{\beta}$ according to Eq. (4).

In conclusion, ELM is an extraordinary simple and fast learning algorithm which needs only one iteration during the learning processing. Compared with the traditional BP algorithm, ELM is capable of initializing randomly the input connection weights and thresholds of hidden nodes. Besides, the number of hidden nodes is the only parameter which needs to be adjusted in ELM algorithm, and it can be determined by constantly updating in practical applications. Substantial experiment on the standard UCI date sets indicates that the ELM algorithm learns fast and provides better generalization performance than the traditional BP algorithm and the model of support vector machine.

Proposed SELM

In this section, firstly, in order to simplify data sets and improve the generalization ability of SELM algorithm, rough set theory (RST) acts as the pre-treatment cell of SELM algorithm to eliminate the redundant attributes of data sets. Then, AP clustering is used to determine the number of hidden nodes and construct Gaussian function as activation functions of hidden nodes, which are transferred to ELM algorithm to improve its self-adaptive ability.

Attribute Reduction of Data Sets

In most of data sets, there are some attributes which do not influence the classification results. So, these attributes can be removed from the data sets. But, how to find these attributes? Fortunately, RST is a useful way. RST introduced by Pawlak [21] is a mathematical tool to deal with vagueness and uncertainty of information. It has proved to be a powerful tool for uncertainty, and it has been applied to data reduction, rule extraction, data mining and granularity computation [22]. RST applies the unclear relation and data pattern comparison based on the concept of an information system with indiscernible data, where the data are uncertain or inconsistent [23].

In RST, redundancy attributes are reduced through the algorithm of attribute reduction (AR). The procedure of AR is as the follows. Firstly, the attributes of data sets are divided into two parts: condition attributes and decision attributes. Then, for each attribute of condition attributes, mark it when it does not influence the classification of data sets according to the condition attributes except it and decision attributes. After that, eliminate all of the marked condition attributes. Finally, the rest attributes and decision attributions compose the reduced data sets.

Determine the Number of Hidden Nodes

In ELM algorithm, the number of hidden nodes is usually determined randomly or artificially, which leads that ELM algorithm cannot develop its greatest ability. So, how to automatically determine the number of hidden nodes is very important. Similar to radial basis function neural

network (RBFNN), the performance of ELM algorithm critically replies on the selection of the number of hidden nodes. In RBFNN, K-means clustering, fuzzy K-means clustering and hierarchical clustering are the conventional strategies to determine the RBF centers of neurons in hidden layer. In our early work [1], AP clustering was used to determine the number, centers and their widths of RBF units which proved to be a more powerful method than above clustering methods. So, in this paper, AP clustering is adopted again to determine the number of hidden nodes in SELM algorithm.

AP clustering, proposed by Frey and Dueck [24], is an efficient approach based on message passing and has been successfully applied to various areas such as clustering images of faces, detecting genes and searching optimal airlines. AP clustering does not require priori knowledge. It simultaneously considers all sample points as potential clustering centers in the beginning, which are called cluster representatives or "exemplars" in the algorithm. AP carries out clustering according to the similarity between N samples and typically takes Euclidean distance as the measurement of similarity between any pair of sample points, which is set as a negative square of the distance. The similarity between any pair of N data points forms $N \times N$ similarity matrix. For the data point x_i and x_j, the similarity $s(i, k)$ is calculated as Eq. (5):

$$s(i,j) = -\left\|x_i - x_j\right\|^2 \tag{5}$$

AP clustering utilizes $s(i, k)$ to indicate how well the data point x_j is suited to be the exemplar for the date point x_i. Each value of $s(k, k)$ on the diagonal of similarity matrix S, called preference p, denotes the evidence for how well suited the date point x_k is to serve as the exemplar. The raise of the value of p will increase the possibility that point x_k becomes an exemplar. Preference p should be initially set equivalent to each other since AP considers that all data points have identical possibilities to be an exemplar. As is well known, preference p influences directly the number of clustering. Smaller value of p results in a smaller number of clusters. And contrarily, larger value of that results in a larger number of clusters. In practical application, the value of p is ordinarily set as the median of the similarity matrix, which results in a moderate number of clusters.

The algorithm determining the number of hidden nodes based on AP clustering is as the follows.

Step 1: Initialization. Initialize the similarity matrix S according to the similarity between data points and set the value of p as the median of S. Define the maximum number of iterations $MaxN$ and set the iteration variable t as zero.

Step 2: Calculate for every data point $k(k = 1, 2 \ldots, N)$ according to the procedures of steps 3–5.

Step 3: Calculate the responsibility and availability of data points according to the following equations:

$$R(i,k) = S(i,k) - \max_j\{A(i,j) + S(i,j)\} \quad j = 1,2,\ldots,N \text{ but } j \neq i,k \tag{6}$$

$$A(i,k) = \min\left\{0, R(k,k) + \sum_j \max(0, R(j,k))\right\} \quad j = 1,2,\ldots,N \text{ but } j \neq i,k \tag{7}$$

$$R(k,k) = P(k) - \max_j\{A(k,j) + S(k,j)\} \quad j = 1,2,\ldots,N \text{ but } j \neq k \tag{8}$$

Step 4: Evaluation. Judge if point k is the center of clustering, using the Eq. (9):

$$R(k,k) + A(k,k) > 0 \tag{9}$$

Step 5: Iteratively update $R(i, k)$ and $A(i, k)$ as follows:

$$R(i+1,k) = (1 - \text{lam}) \cdot R(i,k) + \text{lam} \cdot R(i-1,k) \tag{10}$$

$$A(i+1,k) = (1 - \text{lam}) \cdot A(i,k) + \text{lam} \cdot A(i-1,k) \tag{11}$$

where *lam* is the damping factor of AP clustering, whose value is 0.5.

Step 6: Refresh t as $t + 1$. If the value of t reaches the maximum number of iterations or terminal conditions are satisfied, output the number of clusters h, the centers $c_i(i = 1, 2, \ldots, h)$ of clusters and their widths $\sigma_i(i = 1, 2, \ldots, h)$. Otherwise, it is turned to Step 2.

Activation Function of Hidden Nodes

In ELM algorithm, activation functions mainly include *sigmoid* function, *sine* function, *signum* function and *radial basis* function, which are expressed by Eqs. (12)–(15), respectively [8].

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{12}$$

$$f(x) = \sin(x) \tag{13}$$

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{14}$$

$$f(x) = \mathrm{e}^{-x^2} \tag{15}$$

These functions guarantee the effective operations of ELM algorithm. However, they do not utilize some hidden information of data sets. According to the results of AP clustering, in SELM algorithm, Gaussian basis function is constructed and adopted as activation functions of hidden nodes, which is as follows.

$$\phi_i(x) = \exp\left(-\frac{\|x - c_i\|^2}{2\sigma_i^2}\right) \quad i = 1, 2, \ldots, h \tag{16}$$

where $\|x - c_i\|$ is the Euclidean distance between the input vector x and the center c_i which is one of the outputs of AP clustering and σ_i represents the width of the ith node in the hidden layer. Gaussian function, which has the property of noise suppression, is bounded, strictly positive and continuous [1].

Description of SELM Algorithm

The algorithm of SELM is as follows:

Step 1: Attribute reduction. Utilize the attribute reduction algorithm to reduce data sets, so as to remove their redundant attributes.
Step 2: AP clustering. Cluster the training samples with AP algorithm and calculate the centers and widths of clustering, which are used to be the centers and widths of the activation function in the hidden layer in SLFNs.
Step 3: Calculate the number of clusters and determine automatically the number of hidden node $\tilde{N}$ in SLFNs according to it. Randomly assign hidden node bias $b_i (i = 1, 2, \ldots, \tilde{N})$.
Step 4: Construct Gaussian function with the centers and widths of clustering and consider it as the activation function in hidden layer. Evaluate the hidden layer output matrix H with reduced training samples.
Step 5: Compute the output weight β according to Eq. (4).
Step 6: Calculate the testing error and accuracy of trained SLFNs with testing samples.

Experiments and Analyses

Description and Reduction of Data Sets

The selected data sets from UCI machine learning repository include Iris, Wine, Zoo, Heart Disease, Ionosphere, Cleveland, Wisconsin Breast Cancer Diagnostic (abbreviated to WBCD) and Pima Indian Diabetes (abbreviated to Pima) data sets. The basic characteristics of those data sets and the number of reduced attributes are given in Table 1.

Experiments for Determining the Number of Hidden Nodes

In this section, the experiments for determining the number of hidden nodes of SLFNs are presented. When ELM algorithm is applied, the number of hidden nodes is the only parameter required to be artificially designed. Generally, the number of hidden nodes is gradually increased in a relatively large range in order to evaluate the generalization performance of the networks, and finally the nearly optimal numbers of hidden nodes for each different data set are selected. Figures 1 and 2 present the classification accuracy of the trained networks when the number of hidden nodes varies from 1 to 50.

As given in Table 2, the nearly optimal numbers of hidden nodes for each data set are selected in accordance with Figs. 1 and 2 when the curves of the classification accuracy are relatively stable.

On the other hand, we cluster the reduced training samples through AP clustering. The numbers of clusters with different values of P are presented in Table 3. From the comparison of Tables 2 and 3, we can find that the numbers of clusters through AP clustering are quite close to their manually selected numbers of hidden nodes given in Table 2 if the preference P is set as median(s)/2 for the sizes of the first four data sets are relatively small. For numbers of clusters of the latter four data sets given in Table 3, however, for the sizes of them being relatively large, the numbers of clusters are near to their manually selected numbers of hidden nodes given in Table 2 if the preference P is set as median(s). The above conclusions will be verified further in our subsequent experiments.

Experimental Results of ELM Algorithm and SELM algorithms

In this section, to each data set, we utilize ELM algorithm to train and test the SLFNs with the corresponding selected numbers of hidden nodes shown in section "Experiments for Determining the Number of Hidden Nodes." The activation functions of ELM and SELM are all Gaussian function. For ease of comparison, we compare the ELM and SELM algorithms in the aspects of time expenditure and classification accuracy in order to evaluate the feasibility and reliability of SELM algorithm. The experimental results of ELM and SELM algorithms for the eight data sets are presented in Tables 4 and 5, respectively.

Table 4 shows that there is little difference between training and testing time of the two algorithms while they have the same number of nodes in the hidden layer. The ELM algorithm spends time slightly less than SELM algorithm for the AR algorithm in the SELM algorithm has little time consumption.

From Table 5, we find that the training accuracy of the SELM algorithm is higher than that of the ELM algorithm on most data sets except the Wine and Heart data sets and that all testing accuracy of the SELM algorithm is also higher than that of ELM algorithm except the Wine data sets. Those results indicate that SELM algorithm provides

Table 1 Characteristics of data sets

Data sets	Number of samples		Number of condition attributes	Number of decision attributes	Number of reduced attributes
	Training samples	Testing samples			
Iris	75	75	5	1	4
Wine	89	89	13	1	2
Heart	135	135	13	1	4
Zoo	107	107	16	1	6
Ionosphere	175	176	34	1	3
Cleveland	151	152	13	1	4
WBCD	234	235	30	1	2
Pima	384	384	8	1	3

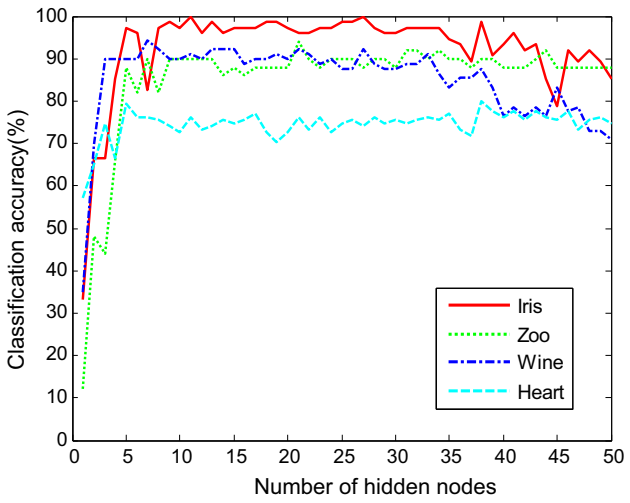

Fig. 1 Classification results of ELM with different numbers of hidden nodes based on the first four data sets

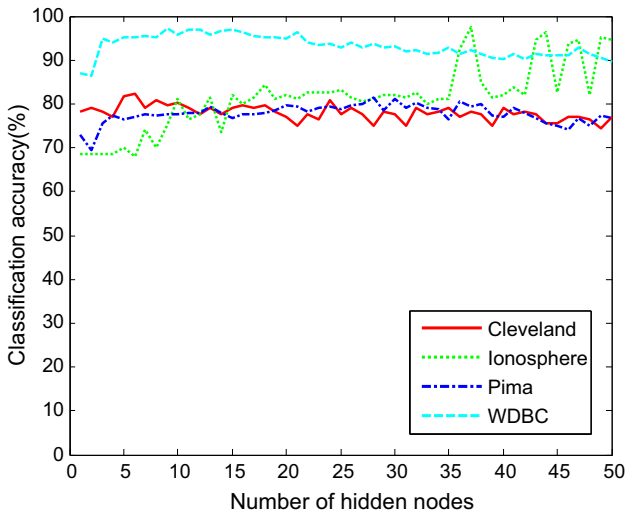

Fig. 2 Classification results of ELM with different numbers of hidden nodes based on the last four data sets

better generalization performance and higher reliability than ELM algorithm.

Experimental Results of SELM Algorithm

For ease of comparison and analysis, the data sets employed in previous experiments are still utilized in the experiments in this section. After AP clustering, we can calculate the number, centers and widths of clusters and use these parameters to set the number of hidden nodes and the centers and widths of activation functions in the SELM algorithm. Next, we compute the hidden layer output matrix H of all training samples through the Gaussian activation functions, train the networks with the SELM algorithm and evaluate the connection weights connecting the hidden layer and the output layer. Finally, we test the trained network using reduced testing samples. The results of training and testing in SELM algorithm are given in Tables 6 and 7, respectively.

Table 6 shows the comparison of training and testing time of SELM algorithm using different values of P. Overall, Table 6 indicates that SELM algorithm spends little training and testing time under any circumstance. The number of clustering centers through AP clustering will increase along with the increment of the value of P, and the large values result in a bit more training time than the small values do, while the testing time differs little. Comparing the training and testing time of ELM algorithm in Table 4, we can find that there is little difference between the training and testing time of the ELM algorithm and the SELM algorithm.

Table 7 presents the training and testing accuracy of the SELM algorithm using different values of P. From Table 7, we find that when the sizes of data sets (the first

Table 2 Numbers of hidden nodes for eight data sets after testing

Data sets	Iris	Wine	Heart	Zoo	Ionosphere	Cleveland	WBCD	Pima
Numbers of hidden nodes	12	15	15	10	47	15	12	15

Table 3 Numbers of clusters through AP clustering using different values of P

Data sets	P = median(s)/2	P = median(s)	P = median(s) * 2
Iris	10	7	5
Wine	15	9	7
Heart	16	8	7
Zoo	11	8	8
Ionosphere	80	43	10
Cleveland	34	15	12
WBCD	49	30	15
Pima	43	29	20

four data sets) are a bit small, the training accuracy and testing accuracy of SELM algorithm are relatively high if the value of P is set as median(s)/2. The reason may be that fewer sizes of data sets may lead to fewer sample instances with similar patterns. It is more appropriate to select more clustering centers to represent the original data sets. However, when the sizes of data sets (the latter four data sets) are a bit large, the training accuracy and testing accuracy of the SELM algorithm are relatively high if the value of P is set as median(s). The reason may be that larger sizes of data set will lead to relatively more sample instances with similar patterns. It is more appropriate to choose moderate number of clustering centers to represent the original data sets. However, overly small number of clustering centers will affect the classification performance of the SLFNs.

The above results validate the previous analysis in section "Experiments for Determining the Number of Hidden Nodes." The number of clustering centers automatically obtained through AP clustering in our experiments is basically equivalent to the nearly optimal number of hidden nodes obtained through manually testing in section "Experiments for Determining the Number of Hidden Nodes," which also confirms the feasibility and

Table 4 Comparison of training and testing time of ELM and SELM

Data sets	ELM		SELM	
	Training time	Testing time	Training time	Testing time
Iris	0.0625	0.0313	0.0625	0.0313
Wine	0.0938	0.0156	0.0938	0.0156
Heart	0.0938	0.0313	0.0469	0.0156
Zoo	0.0625	0.0313	0.0625	0.0313
Ionosphere	0.0938	0.0313	0.0625	0.0313
Cleveland	0.0625	0.0313	0.0625	0.0313
WBCD	0.0781	0.0313	0.0469	0.0313
Pima	0.0938	0.0313	0.0625	0.0156

Table 5 Comparison of training and testing accuracy of ELM and SELM

Data sets	ELM		SELM	
	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
Iris	96	94.67	97.33	97.33
Wine	100	94.38	88.76	91.01
Heart	83.7	75.56	80.74	76.3
Zoo	90	82	96	90
Ionosphere	86.86	86.29	96	94.15
Cleveland	82.12	77.48	83.44	80.79
WBCD	95.42	92.61	95.42	94.72
Pima	78.125	77.08	80.45	77.34

Table 6 Comparison of training and testing times of SELM using different values of P

Data sets	P = median(s)/2		P = median(s)		P = median(s) * 2	
	Training time	Testing time	Training time	Testing time	Training time	Testing time
Iris	0.0781	0.0313	0.0625	0.0313	0.0781	0.0313
Wine	0.0469	0.0313	0.0469	0.0313	0.0625	0.0313
Heart	0.0938	0.0313	0.0781	0.0313	0.0938	0.0313
Zoo	0.0625	0.0313	0.0625	0.0313	0.0313	0.0313
Ionosphere	0.0938	0.0625	0.0625	0.0313	0.0625	0.1563
Cleveland	0.0625	0.0469	0.0625	0.0313	0.0625	0.1563
WBCD	0.0938	0.0625	0.0781	0.0313	0.0625	0.0313
Pima	0.0938	0.0625	0.0625	0.0625	0.0313	0.1094

Table 7 Comparison of training and testing accuracy of SELM using different values of P

Data sets	P = median(s)/2		P = median(s)		P = median(s) * 2	
	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy	Training accuracy	Testing accuracy
Iris	97.33	93.33	96	92	89.33	93.33
Wine	100	97.75	94.38	91.01	92.13	86.52
Heart	86.67	85.19	83.7	82.22	82.96	80.74
Zoo	100	96	94	76	86	82
Ionosphere	98.86	82.86	100	94.29	85.71	85.71
Cleveland	84.11	80.79	85.43	81.45	83.44	80.13
WBCD	97.18	95.77	96.48	96.13	96.83	94.37
Pima	79.37	78.59	80.94	79.11	78.85	77.81

reliability of AP clustering when it is applied to optimize ELM algorithm.

Conclusions

In this paper, an improved ELM algorithm called the SELM algorithm self-adaptively determines the number of the hidden nodes during learning by AP clustering to find the optimal architecture of network. A performance comparison of the SELM with the traditional ELM has been carried out on some typical benchmark classification data sets from UCI machine learning repository. To be precise, the SELM algorithm achieves competitive testing accuracy at significantly lower training time on the benchmark classification data sets. Further, we also test and verify the availability of attribute reduction and the influences on the SELM algorithm when three different values of P were adopted in AP clustering. In future work, we will improve the accuracy of the SELM algorithm by regularization methods and statistical learning theory. Additionally, more practical applications will be also studied to verify the availability of the SELM algorithm.

Acknowledgments This work is supported by the Fundamental Research Funds for the Central Universities (No. 2015XKMS088).

Compliance with Ethical Standards

Conflict of Interest Li Xu, Shifei Ding, Xinzheng Xu and Nan Zhang declare that they have no conflict of interest.

Informed Consent All procedures followed were in accordance with the ethical standards of the responsible committee on human experimentation (institutional and national) and with the Helsinki Declaration of 1975, as revised in 2008 (5). Additional informed consent was obtained from all patients for which identifying information is included in this article.

Human and Animal Rights This article does not contain any studies with human or animal subjects performed by the any of the authors.

References

1. Xin-Zheng X, Ding S-F, Shi Z-Z, Zhu H. Optimizing radial basis function neural network based on rough set and AP clustering algorithm. J Zhejiang Univ (SCIENCE A). 2012;13(2):131–8.
2. He Q, Shang T-F, Zhuang F-Z, Shi Z-Z. Parallel extreme learning machine for regression based on MapReduce. Neurocomputing. 2013;102(2):52–8.

3. Xie F-Y, Bovik VAC. Automatic segmentation of dermoscopy images using self-generating neural networks seeded by genetic algorithm. Pattern Recogn. 2013;46(3):1012–9.
4. Ding S-F, Jia W-K, Chun-Yang S, et al. Research of neural network algorithm based on factor analysis and cluster analysis. Neural Comput Appl. 2011;20(2):297–302.
5. Razavi S, Tolson BA. A new formulation for feedforward neural networks. IEEE Trans Neural Netw. 2011;22(10):1588–98.
6. Kimura D, Nii M, Yamaguchi T, Takahashi Y, Yumoto T. Fuzzy nonlinear regression analysis using fuzzified neural networks for fault diagnosis of chemical plants. J Adv Comput Intell Intell Inf. 2011;15(3):336–44.
7. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004). Hungary: Budapest; 2004. 985–90.
8. Huang G-B, Zhu QinYu, Siew C-K. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1–3): 489–501.
9. Huang G-B, Chen L, Siew C-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw. 2006;17(4):879–92.
10. Huang G-B. An Insight into Extreme learning machines: random neurons, random features and kernels. Cogn Comput. 2014;6(3): 376–90.
11. Ding S-F, Xin-Zheng X, Nie R. Extreme learning machine and its applications. Neural Comput Appl. 2014;25(3):549–56.
12. Huang G, Huang G-B, Song S-J, You K-Y. Trends in extreme learning machines: A review. Neural Netw. 2015;61:32–48.
13. Kasun L-L-C, Zhou H-M, Huang G-B, Vong C-M. Representational learning with extreme learning machine for big data. IEEE Intell Syst. 2013;28(6):31–4.
14. Bai Z, Huang G-B, Wang D-W, Wang H, Westover M-B. Sparse extreme learning machine for classification. IEEE Trans Cybern. 2014;44(10):1858–70.
15. Huang G, Song S-J, Gupta J-N-D, Wu C. Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybern. 2014;44(12):2405–17.
16. Lin S-B, Liu X, Fang J, Zong-Ben X. Is extreme learning machine feasible? A theoretical assessment (part II). IEEE Trans Neural Netw Learn Syst. 2015;26(1):21–34.
17. Rong H-J, Ong Y-S, Tan A-H, Zhu Z-X. A fast pruned-extreme learning machine for classification problem. Neurocomputing. 2008;72(1–3):359–66.
18. Huang G-B, Li M-B, Chen L, Siew C-K. Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing. 2008;71(4–6):576–83.
19. Huang G-B, Chen L. Convex incremental extreme learning machine. Neurocomputing. 2007;70(16–18):3056–62.
20. Huang G-B, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. 2008;71(16–18): 3460–8.
21. Pawlak Z. Rough set. Int J Comput Inf Sci. 1982;11:341–56.
22. Cao Y, Chen X-H, Wu DD, Mo M. Early warning of enterprise decline in a life cycle using neural networks and rough set theory. Exp Syst Appl. 2011;38(6):6424–9.
23. Cheng J-H, Chen H-P, Lin Y-M. A hybrid forecast marketing timing model based on probabilistic neural network, rough set and C4.5. Exp Syst Appl. 2010;37(3):1814–20.
24. Frey B, Dueck D. Clustering by passing messages between data points. Science. 2007;315(5814):972–6.