Neural Comput & Applic
DOI 10.1007/s00521-015-1874-3

Self-adaptive extreme learning machine

Gai-Ge Wang[1,2,3] · Mei Lu[1] · Yong-Quan Dong[1] · Xiang-Jun Zhao[1]

Received: 14 September 2014 / Accepted: 9 March 2015

Abstract In order to overcome the disadvantage of the traditional algorithm for SLFN (single-hidden layer feedforward neural network), an improved algorithm for SLFN, called extreme learning machine (ELM), is proposed by Huang et al. However, ELM is sensitive to the neuron number in hidden layer and its selection is a difficult-to-solve problem. In this paper, a self-adaptive mechanism is introduced into the ELM. Herein, a new variant of ELM, called self-adaptive extreme learning machine (SaELM), is proposed. SaELM is a self-adaptive learning algorithm that can always select the best neuron number in hidden layer to form the neural networks. There is no need to adjust any parameters in the training process. In order to prove the performance of the SaELM, it is used to solve the Italian wine and iris classification problems. Through the comparisons between SaELM and the traditional back propagation, basic ELM and general regression neural network, the results have proven that SaELM has a faster learning speed and better generalization performance when solving the classification problem.

✉ Gai-Ge Wang
gaigewang@163.com; gaigewang@gmail.com

Mei Lu
lumei@jsnu.edu.cn

Yong-Quan Dong
tomdyq@163.com

Xiang-Jun Zhao
xjzhao@jsnu.edu.cn

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

[2] Institute of Algorithm and Big Data Analysis, Northeast Normal University, Changchun 130117, China

[3] School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China

Keywords Classification · Self-adaptive · Extreme learning machine · Back propagation · General regression neural network

1 Introduction

Due to its good learning ability, SLFN has been widely used in many areas. However, the traditional learning algorithms (such as BP algorithm) have some inherent shortcomings. These disadvantages have become a major bottleneck that restricts its further development. In most cases, the gradient descent method is used in feedforward neural networks (FNN), which has the following main disadvantages.

1. Many iterations are required in the gradient descent method in order to adjust weights and thresholds. Therefore, the training process takes a long time.
2. It is easy to fall into local minimum, so that it cannot achieve the global minimum.
3. It is very sensitive to the choice of learning rate η. Due to high influence on the performance for neural networks (NN), proper learning rate η must be selected in order to obtain a more ideal network. If η is too small, then convergent speed of the algorithm is very slow, and the training process would take a long time; conversely, if η is too big, the training process may be unstable and it is hard to converge to the global optimum.

Therefore, in order to enhance the performance of FNN, many scholars are always striving for exploring a training algorithm that has a fast training speed, a global optimal solution and a good generalization performance. Therefore, an improved algorithm for SLFN, called extreme learning machine (ELM), is proposed by Huang et al. [1]. The algorithm randomly generates the connection weights and thresholds between input layer and hidden layer in the training process. However, ELM is sensitive to the number of neurons in hidden layer that is difficult to solve. In this paper, by combination of the self-adaptive mechanism and ELM, an improved ELM, called SaELM, is proposed. SaELM can always select the best neuron number in hidden layer to construct the optimal networks. Therefore, there are no parameters to adjust in SaELM. Most importantly, a unique optimal solution can be obtained all the time. In order to prove the performance of the SaELM, it is used to solve Italian wine and iris classification problems. By comparing with the traditional BP, ELM and general regression neural network (GRNN), this method has a faster learning speed and better generalization performance.

The remainder of this paper is organized as follows. The next two sections describe the preliminary of ELM and its related works, respectively. Sections 4 and 5 represent the framework of SaELM and the detailed classification by using SaELM. Then, in Sect. 6, a series of comparison experiments on Italian wine and iris classification problems are conducted. The final section provides our concluding remarks and points out our future work orientation.

2 ELM

2.1 The idea of ELM

A typical SLFN consists of input layer, hidden layer and output layer, and the relationship between input layer and hidden layer and between hidden layer and output layer neurons is fully connected, as shown in Fig. 1. Here, the input layer has n neurons, responding to n input variables; hidden layer has l neurons; and output layer has m neurons, corresponding to m output variables. Without loss of generality, we assume the weights between input layer and the hidden layer $\boldsymbol{\omega}$ is as follows:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2n} \\ \vdots & \vdots & & \vdots \\ \omega_{l1} & \omega_{l2} & \cdots & \omega_{\text{ln}} \end{bmatrix}_{l\times n} \quad (1)$$

where ω_{ji} is the weight between neuron i and neuron j in the hidden layer.

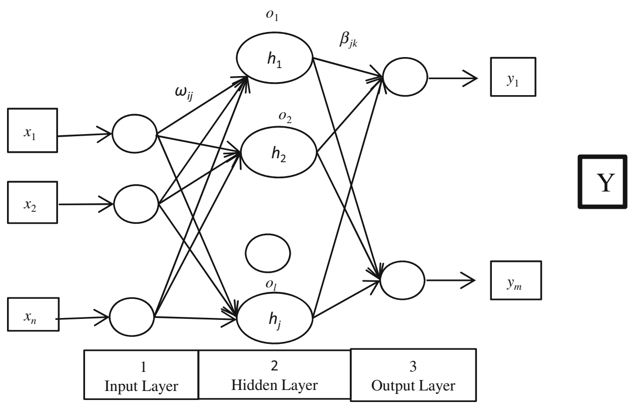

Fig. 1 Topology of SLFN

Assume connection weight between the hidden layer and output layer $\boldsymbol{\beta}$ is as follows:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \vdots & \vdots & & \vdots \\ \beta_{l1} & \beta_{l2} & \cdots & \beta_{lm} \end{bmatrix}_{l\times m} \quad (2)$$

where β_{jk} represents the weight between the jth neuron in hidden layer and kth neuron in output layer.

Let neuron threshold in the hidden layer $\mathbf{b}$ be

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_l \end{bmatrix}_{l\times 1} \quad (3)$$

Assume training set has Q samples, and its input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$ can be given as:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1Q} \\ x_{21} & x_{22} & \cdots & x_{2Q} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nQ} \end{bmatrix}_{n\times Q}, \quad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1Q} \\ y_{21} & y_{22} & \cdots & y_{2Q} \\ \vdots & \vdots & & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mQ} \end{bmatrix}_{m\times Q} \quad (4)$$

Let activation function in the hidden layer be $g(x)$. According to Fig. 1, the output of the network $\mathbf{T}$ can be defined as:

$$\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_Q]_{m\times Q},\ \mathbf{t}_j = \begin{bmatrix} t_{1j} \\ t_{2j} \\ \vdots \\ t_{mj} \end{bmatrix}_{m\times 1} = \begin{bmatrix} \sum_{i=1}^{l} \beta_{i1} g(w_i x_j + b_i) \\ \sum_{i=1}^{l} \beta_{i2} g(w_i x_j + b_i) \\ \vdots \\ \sum_{i=1}^{l} \beta_{im} g(w_i x_j + b_i) \end{bmatrix}_{m\times 1},\ j = 1, 2, \ldots, Q \quad (5)$$

where $\boldsymbol{\omega}_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{in}]$, $\mathbf{x}_j = [x_{j1}, x_{j2}, \ldots, x_{nj}]^{\mathrm{T}}$.

Equation (5) can be written as:

$$\mathbf{H}\beta = \mathbf{T}' \quad (6)$$

where $\mathbf{T}'$ is the transpose of matrix $\mathbf{T}$; $\mathbf{H}$ is the output matrix of hidden layer, and it can be represented as:

$$H(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_l, b_1, b_2, \ldots, b_l, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_Q) = \begin{bmatrix} g(\boldsymbol{\omega}_1\mathbf{x}_1 + b_1) & g(\boldsymbol{\omega}_2\mathbf{x}_1 + b_2) & \cdots & g(\boldsymbol{\omega}_l\mathbf{x}_1 + b_l) \\ g(\boldsymbol{\omega}_1\mathbf{x}_2 + b_1) & g(\boldsymbol{\omega}_2\mathbf{x}_2 + b_2) & \cdots & g(\boldsymbol{\omega}_l\mathbf{x}_2 + b_l) \\ \vdots & \vdots & & \vdots \\ g(\boldsymbol{\omega}_1\mathbf{x}_Q + b_1) & g(\boldsymbol{\omega}_2\mathbf{x}_Q + b_2) & \cdots & g(\boldsymbol{\omega}_l\mathbf{x}_Q + b_l) \end{bmatrix}_{Q\times l} \quad (7)$$

On the basis of previous studies, Huang et al. proposed the following two theorems (the detailed theorem can be seen in [2]).

Theorem 1 Given any Q different samples (x_i, t_i), where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\mathrm{T}} \in \mathbf{R}^n$, $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}] \in \mathbf{R}^m$, an arbitrary interval infinitely differentiable activation function g: $\mathbf{R} \to \mathbf{R}$, then for a SLFN with Q neurons in hidden layer, in any assignment $\boldsymbol{\omega}_i \in \mathbf{R}^n$ and $b_i \in \mathbf{R}$ cases, the output matrix in hidden layer $\mathbf{H}$ is reversible and there is $\|\mathbf{H}\beta - \mathbf{T}'\| = 0$.

Theorem 2 Given any Q different samples (x_i, t_i), where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\mathrm{T}} \in \mathbf{R}^n$, $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}] \in \mathbf{R}^m$, given arbitrarily small error ε ($\varepsilon > 0$) and an arbitrary interval infinitely differentiable activation function g: $\mathbf{R} \to \mathbf{R}$, then there is always a SLFN with K ($K \le Q$) neurons in hidden layer, in any assignment $\boldsymbol{\omega}_i \in \mathbf{R}^n$ and $b_i \in \mathbf{R}$ cases, and there is $\|\mathbf{H}_{n\times M}\boldsymbol{\beta}_{M\times n} - \mathbf{T}'\| < \varepsilon$.

According to Theorem 1, if the number of neurons in hidden layer and the number of training samples are equal, then for any $\boldsymbol{\omega}$ and $\mathbf{b}$, SLFN can approach to training samples without error, i.e.,

$$\sum_{j=1}^{Q} \|\mathbf{t}_j - \mathbf{y}_j\| = 0 \quad (8)$$

where $\mathbf{y}_j = [y_{1j}, y_{2j}, \ldots, y_{mj}]^{\mathrm{T}}$, $j = 1, 2, \ldots, Q$.

However, when the number of samples in training set Q is a little big, the number of neurons in hidden layer K is usually smaller than Q in order to reduce the amount of calculation. According to Theorem 2, the training error of SLFN can approximate an arbitrary number $\varepsilon > 0$, that is

$$\sum_{j=1}^{Q} \|\mathbf{t}_j - \mathbf{y}_j\| < \varepsilon \quad (9)$$

Therefore, when the activation function $g(x)$ is infinitely differentiable, there is no need to adjust all the parameters in SLFN. $\mathbf{w}$ and $\mathbf{b}$ can be randomly generated before training, and they can remain unchanged in the training process. And the connection weight $\boldsymbol{\beta}$ between hidden layer and output layer can be obtained by the least squares solution of the following equation:

$$\min_{\boldsymbol{\beta}} \|\mathbf{H}\beta - \mathbf{T}'\| \quad (10)$$

Its solution is

$$\hat{\beta} = \mathbf{H}^{+}\mathbf{T}' \quad (11)$$

where $\mathbf{H}^{+}$ is the Moore–Penrose generalized inverse of the output matrix $\mathbf{H}$ in hidden layer.

2.2 Learning algorithm in ELM

According to Sect. 2.1, in ELM, $\mathbf{w}$ and $\mathbf{b}$ are randomly generated before training. We simply determine the number of neurons and activation function (infinitely differentiable) in hidden layer, and then, $\boldsymbol{\beta}$ can be calculated. Specifically, the main steps of the ELM learning algorithm can be represented as follows:

1. The number of neurons in hidden layer is set, and the connection weight $\mathbf{w}$ and the bias $\mathbf{b}$ between input layer and hidden layer are randomly generated.
2. An infinitely differentiable function is considered as neuron activation function in the hidden layer, and then, the output matrix $\mathbf{H}$ of the hidden layer is calculated.
3. The weight of the output layer $\hat{\beta}$ is calculated as follows: $\hat{\beta} = \mathbf{H}^{+}\mathbf{T}'$

It is worth mentioning that the previous researches have shown that not only many nonlinear activation functions can be used in ELM (such as the S function, sine function and composite functions), but also non-differentiable function as the activation function.

3 Related works

Huang et al. [3] surveyed ELM and its variants, especially on (1) batch learning mode of ELM, (2) fully complex ELM, (3) online sequential ELM, (4) incremental ELM and

(5) ensemble of ELM. Some of other representative works about ELM can be described as follows.

In order to overcome the computational and outlier robustness problems in ELM, Horata et al. [4] proposed an extended complete orthogonal decomposition (ECOD) method to solve the computational problem in ELM weights computing via ECODLS algorithm. Yang et al. [5] proposed a new learning algorithm, called bidirectional ELM (B-ELM), in which some hidden nodes are not randomly selected. Equivalence between ELM and the positive and negative fuzzy rule system is revealed in [6], and ELM is then used for training the positive and negative fuzzy rule system quickly for image classification. Pouzols et al. [7] proposed a kind of ELM, called evolving fuzzy optimally pruned ELM (eF-OP-ELM) that is followed by a random projection-based approach to extracting evolving fuzzy rulebases. Li et al. [8] proposed an enhanced ELM based on ridge regression (RR-ELM) for regression, which replaces the least square method to calculate output weights. In [9], a dynamic ensemble ELM based on sample entropy is proposed, which can alleviate to some extent the problems of instability and over-fitting and increase the prediction accuracy. To deal with data with imbalanced class distribution, a weighted ELM [10] is proposed which is able to generalize to balanced data. Through analyzing the mechanism of ELM algorithm, He et al. [11] designed a parallel ELM for regression by using MapReduce framework, which is a current simple but powerful parallel programming technique. Wang et al. [12] provided an evolutionary approach for constituting ELM ensembles, which employed the model diversity as fitness function to direct the selection of base learners. Feng et al. [13] proposed an efficient approach to determine the number of hidden nodes in ELM.

In application, Wang et al. [14] used ELM to solve image deblurring problem, and this method can remove noise and protect edge. Iosifidis et al. [15] proposed a novel method that performs dynamic action classification by exploiting the effectiveness of the ELM algorithm for single-hidden layer FNN training. Based on optimized sample entropy and ELM, Song et al. [16] proposed a novel method to perform automatic epileptic seizure detection in EEGs. Chacko et al. [17] dealt with the recognition of handwritten Malayalam character using wavelet energy feature (WEF) and ELM. Text categorization was solved by a novel approach based on a regularization ELM (RELM) in which its weights can be obtained analytically, and a bias–variance trade-off could be achieved by adding a regularization term into the linear system of single-hidden layer FNN [18]. In [19], a multistage ELM is proposed to improve the accuracy of clustering. During the process of this new ELM, the input data were divided into several stages; then, every stage was analyzed independently. Finally, this method has been used in hydraulic tube tester data. Suresh et al. [20] proposed a machine learning approach to measure the visual quality of JPEG-coded images, in which the problem of quality estimation is transformed to a classification problem and solved using ELM algorithm. In [21], by combining Gray relation analysis and ELM with Taguchi method, the Gray ELM (GELM) is proposed to help retailers make decisions. Through the combined wavelet transform-ELM (WT-ELM) technique, Malathi et al. [22] proposed a new approach for fault section identification, classification and location in a series-compensated transmission line.

In addition, many scholars have combined ELM with other intelligent optimization algorithms. Han et al. used a modified particle swarm optimization (PSO) [23, 24] algorithm to select the input weights and hidden biases of single-hidden layer SLFN. Cao et al. [29] combined differential evolution (DE) [25–27] with evolutionary ELM [28] and proposed an improved learning algorithm named self-adaptive ELM (SaELM), in which DE is used to optimize the network hidden node parameters. Similarly, some other meta-heuristics will be hybridized with ELM, such as the cuckoo search (CS) [30, 31], firefly algorithm (FA) [32, 33], Grey Wolf optimizer (GSO) [34], harmony search (HS) [35–37], biogeography-based optimization (BBO) [38–42], animal migration optimization (AMO) [43], gravitational search algorithm [44–46], ant colony optimization [47, 48], interior search algorithm (ISA) [49], bat algorithm (BA) [50, 51] and krill herd (KH) [52–55].

4 SaELM

As discussed before, for the basic ELM, the selection of the neuron number in hidden layer N is a difficult problem. In the present work, a self-adaptive mechanism is originally proposed in order to select the optimal neuron number in hidden layer N. By combination of this self-adaptive mechanism and basic ELM, the so-called self-adaptive ELM (SaELM) is proposed that always uses the optimal neuron number in hidden layer N to construct the NN. The mainframe of SaELM can be represented in Algorithm 1.

Algorithm 1 *SAELM algorithm*

```
Begin
    Step 1: Initialization. Set the minimum and maximum of neurons in hidden layer $N_{min}$ and
            $N_{max}$; set the maximal generation MaxIter and current generation counter $G = 1$;
            initialize the interval of neuron number $N_{Interval}$.
    Step 2: While $N_{Interval}$ is not equal to zero do
              Calculate M.
              for j=1:M (all neuron number between $N_{min}$ and $N_{max}$) do
                  While G<MaxIter do
                      Perform the following motion calculation.
                      Construct the basic ELM with $N_j$
                      Train and predict as shown in Section 2.2.
                      Evaluate the prediction results.
                      $G = G+1$.
                  end while
              end for j
              Select the best neuron number $N_{best}$ with minimum error.
              Adjust $N_{min}$, $N_{max}$ and $N_{Interval}$.
    Step 3: end while
    Step 4: Output the best neuron number in hidden layer results and accuracy.
End.
```

In Algorithm 1, $N_{\min}$ and $N_{\max}$ represent the minimum and maximum of neurons in hidden layer. N_{Interval} is the interval of neuron number. That is to say, $N_1 = N_{\min}$, $N_2 = N_{\min} + N_{\text{Interval}}$, $N_3 = N_{\min} + 2 \times N_{\text{Interval}}, \ldots,$ $N_j = N_{\min} + (j-1) \times N_{\text{Interval}}$, …, $N_M = N_{\max}$ ($j = 1, 2, \ldots, M$). Clearly, M can be calculated as follows:

$$M = \text{floor}\left(\frac{N_{\max} - N_{\min}}{N_{\text{Interval}}}\right) + 1 \tag{12}$$

where floor(x) rounds the elements of x to the nearest integers toward minus infinity.

For each neuron number N_j ($j = 1, 2, \ldots, M$), SaELM is constructed according to the basic ELM as shown in Sect. 2.1. And then, SaELM is trained and is used to predict. In order to get representative performance, each SaELM is run *MaxIter* times. After *MaxIter* generation for each neuron number N_j ($j = 1, 2, \ldots, M$), the best neuron number N_{best} is selected with minimum error. Subsequently, $N_{\min}$, $N_{\max}$ and N_{Interval} are updated according to N_{best}, as shown in Eqs. (13)–(15).

$$N_{\min} = \max(N_{\text{best}} - Q * N_{\text{Interval}}, N_{\min}) \tag{13}$$

$$N_{\max} = \min(N_{\text{best}} + Q * N_{\text{Interval}}, N_{\max}) \tag{14}$$

$$N_{\text{Interval}} = \text{floor}(N_{\text{Interval}}/L) \tag{15}$$

where Q and L are width factor and scale factor, respectively. The width factor Q represents the range of neuron number in hidden layer for the next run. If Q is too big, the range of the neuron number is large, and it has a relative high possibility to find the best neuron number with a slow speed; if Q is too small, the range is small, too, and it is possible not to find the best neuron number though fast. The scale factor L has an influence on the difference between two adjacent neuron numbers. If L is too big, the difference between two adjacent neuron numbers is becoming small sharply, and SaELM stops with a fast speed; if L is small, the difference is big and SaELM stops slowly. It is clear that N_{Interval} should be equal to or bigger than 1. When N_{Interval} is equal to zero, the SaELM stops and outputs the best neuron number N_{best} in hidden layer and its responding accuracy.

5 The SaELM model for classification problem

According to Sect. 3, we can use SaELM to solve classification problem. In the present work, the proposed SaELM is applied to solve classification problem. The detailed step can be represented as follows (see Fig. 2).

1. Preprocessing
   In order to remove the influence of the singular data on the prediction results, the data are firstly normalized.
2. Generating the training set and test set
   Similar to FNN, a sufficient number of representative training samples is required so that the prediction model has good generalization performance. Meanwhile, we must format the training set and test set in order to satisfy the requirements of the SaELM.

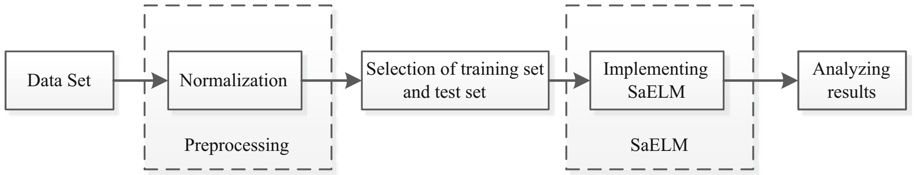

Fig. 2 Flowchart of SaELM for classification

3. Implementing SaELM
   According to Sect. 2, ELM is constructed and then is trained by using training set. It is worth mentioning that, as discussed in Sect. 2, the number of neurons in hidden layer has a great impact on the performance of ELM. By contrast, SaELM can self-adaptively select the optimal neuron number in hidden layer, and ELM can always perform in the best way. Therefore, the only thing we need to do is inputting the data samples into the SaELM. And then, the SaELM can select the best neuron number to form the optimal networks and provide the prediction results.
4. Performance evaluation
   By calculating the error between the predicted value and the true value (mean square error, coefficient of determination, the accuracy), the generalization ability of the model can be evaluated. Meanwhile, run-time used by SaELM can be compared with the traditional FNN and ELM, and thus, the speed of ELM is also evaluated.

6 Simulation results

In this section, in order to show the advantage of the SaELM, SaELM is compared with BP, ELM and GRNN through a classification problem. Firstly, the representation of Italian wine and iris classification problems can be provided in the following section.

6.1 Italian wine classification problem

6.1.1 Problem description

Wine data are the physical chemistry-related fields of data, and its source is originated from UCI wine database (http://archive.ics.uci.edu/ml/datasets/wine). It records three different varieties of wine on the chemical composition analysis grown in the same region in Italy. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The database contains 178 samples, each sample contains 13 feature components (chemical composition), and each sample category label has given. The attributes in each sample can be represented as follows:

1. Alcohol
2. Malic acid
3. Ash
4. Alkalinity of ash
5. Magnesium
6. Total phenols
7. Flavonoids
8. Non-flavonoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

In a classification context, this is a well-posed problem with "well-behaved" class structures.

Different kinds of wine are identified with 1, 2 and 3. Each sample contains 14 dimensions. The first dimension represents a class identifier, and the others represent the characteristics of wine. In these 178 samples, 1–59, 60–130 and 131–178 belong to the first, second and third category, respectively. Each category is divided into two parts: training set and test set.

6.1.2 Performance of SaELM

In order to show the performance of SaELM, here we use it to solve the Italian wine classification problem. The activation function used in SaELM is S function. The parameters used in SaELM are set as follows: width factor $Q = 2$, scale factor $L = 4$, $N_{\min} = 5$, $N_{\max} = 120$, $N_{\text{Interval}} = 5$ and the maximum generation $MaxIter = 2000$. Therefore, M is 24 according to Eq. (12). That is, $N_1 = N_{\min} = 5$, $N_2 = N_{\min} + N_{\text{Interval}} = 10$, $N_3 = N_{\min} + 2 \times N_{\text{Interval}} = 15$, …, $N_j = N_{\min} + (j-1) \times N_{\text{Interval}} = 5 + (j-1) \times 5 = 5 \times j$, …, $N_M = N_{24} = N_{\max} = 120$. The results are given in Table 1 and Fig. 3.

In Table 1, we can see that SaELM reaches the maximal accuracy 97.15 % (86.46/89) when $N_{\text{best}} = 20$. When N reaches 5–20, the SaELM has not learnt enough features to classify the test set. Therefore, the accuracy of test set is becoming much better as the N increases. At the point of about 20, the SaELM approaches the best state, and the accuracy of training set and test set reaches climax. After that, SaELM is in over-fitting state. Therefore, the accuracy of training set is almost unchanged, and the test set is

Table 1 Accuracy of SaELM with $N_{\min} = 5$, $N_{\max} = 120$ and $N_{\text{Interval}} = 5$ for the Italian wine classification problem

N	Training set				Test set			
	Mean	Best	Worst	Std	Mean	Best	Worst	Std
5	74.40	87	50	6.58	73.24	88	35	8.11
10	85.47	888	72	2.23	83.26	89	69	3.45
15	88.42	89	81	0.83	86.17	89	74	1.50
20	88.96	89	88	0.18	**86.46**	**89**	**82**	**1.11**
25	88.97	89	88	0.13	86.30	89	**82**	1.15
30	89.00	89	88	0.04	86.00	89	**82**	1.20
35	89.00	89	88	0.07	85.65	89	76	1.37
40	89.00	89	88	0.11	85.42	89	80	1.56
45	**89**	89	89	0	85.11	89	79	1.78
50	**89**	89	89	0	84.12	89	76	2.06
55	**89**	89	89	0	83.66	89	76	2.38
60	**89**	89	89	0	82.25	89	71	2.87
65	**89**	89	89	0	80.34	88	66	3.22
70	**89**	89	89	0	77.25	87	65	4.12
75	**89**	89	89	0	74.53	86	54	5.34
80	**89**	89	89	0	68.26	86	42	6.21
85	**89**	89	89	0	57.25	81	27	8.56
90	**89**	89	89	0	47.35	73	22	9.12
95	**89**	89	89	0	59.45	76	36	7.34
100	**89**	89	89	0	65.10	82	40	6.45
105	**89**	89	89	0	68.23	83	48	5.65
110	**89**	89	89	0	70.12	83	56	5.23
115	**89**	89	89	0	71.50	84	55	5.14
120	**89**	89	89	0	72.81	84	55	4.66

Best value obtained by each method is in bold

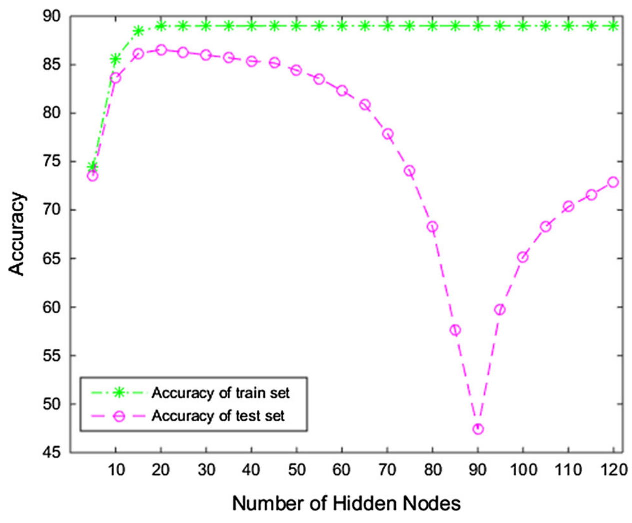

Fig. 3 Prediction of test samples (SaELM) with $N_{\min} = 5$, $N_{\max} = 120$ and $N_{\text{Interval}} = 5$ for the Italian wine classification problem

getting slower. When *N* is equal to 90, it is the lowest. Later, SaELM may learn more from the wine data, and the accuracy of test set is getting higher though lower than the highest accuracy. And then, $N_{\min}$, $N_{\max}$ and N_{Interval} are updated according to Eqs. (13)–(15), as shown in Eqs. (16)–(18).

$$\begin{aligned} N_{\min} &= \max(N_{\text{best}} - Q * N_{\text{Interval}}, N_{\min}) \\ &= \max(20 - 2 \times 5,\ 10) \\ &= 10 \end{aligned} \tag{16}$$

$$\begin{aligned} N_{\max} &= \min(N_{\text{best}} + Q * N_{\text{Interval}}, N_{\max}) \\ &= \min(20 + 2 \times 5,\ 120) \\ &= 30 \end{aligned} \tag{17}$$

$$\begin{aligned} N_{\text{Interval}} &= \text{floor}(N_{\text{Interval}}/L) \\ &= \mathit{floor}(5/4) \\ &= 1 \end{aligned} \tag{18}$$

Therefore, *M* is 21 according to Eq. (12). That is, $N_1 = N_{\min} = 10$, $N_2 = N_{\min} + N_{\text{Interval}} = 11$, $N_3 = N_{\min} + 2 \times N_{\text{Interval}} = 12$, …, $N_j = N_{\min} + (j-1) \times N_{\text{Interval}} = 10 + (j-1) \times 1{=}9 + j$, …, $N_M = N_{21} = N_{\max} = 30$. The results are given in Table 2 and Fig. 4. When *N* reaches 10–18, the training for SaELM is insufficient. Therefore, the accuracy of training set and test set is becoming much better as the *N* increases. When *N* reaches 18, the SaELM is in the best state, and the accuracy of training set and test set reaches climax. After that, SaELM is in over-fitting state, and the accuracy of training set is therefore the highest value and almost unchanged. However, the test set is getting slower as the *N* increases. When *N* is equal to 90, it is the lowest. Later, SaELM may learn more from the wine data, and the accuracy of test set is getting higher though lower than the highest accuracy. Table 2 and Fig. 4 show that more hidden nodes require more computations, but not always lead to good results. In sum, proper number of hidden nodes is the best choice for the SaELM.

In Table 2, we can see that SaELM reaches the maximal accuracy 97.20 % (86.51/89) when $N_{\text{best}} = 18$. And then, $N_{\min}$, $N_{\max}$ and N_{Interval} are updated according to Eqs. (13)–(15), as shown in Eqs. (19)–(21).

$$\begin{aligned} N_{\min} &= \max(N_{\text{best}} - Q * N_{\text{Interval}}, N_{\min}) \\ &= \max(18 - 2 \times 1,\ 10) \\ &= 16 \end{aligned} \tag{19}$$

$$\begin{aligned} N_{\max} &= \min(N_{\text{best}} + Q * N_{\text{Interval}}, N_{\max}) \\ &= \min(18 + 2 \times 1,\ 30) \\ &= 20 \end{aligned} \tag{20}$$

$$\begin{aligned} N_{\text{Interval}} &= \text{floor}(N_{\text{Interval}}/L) \\ &= \text{floor}(1/4) \\ &= 0 \end{aligned} \tag{21}$$

According to Eq. (21), N_{Interval} is equal to zero; therefore, the SaELM stops and returns the best neuron number $N_{\text{best}} = 18$ and its responding accuracy 97.20 % (86.51/89).

Table 2 Accuracy of SaELM with $N_{\min} = 10$, $N_{\max} = 30$ and $N_{\mathrm{Interval}} = 1$ for the Italian wine classification problem

N	Training set				Test set			
	Mean	Best	Worst	Std	Mean	Best	Worst	Std
10	84.49	**89**	**73**	2.38	83.42	**89**	**65**	3.52
11	86.57	**89**	76	1.80	84.39	**89**	69	2.88
12	87.16	**89**	77	1.48	84.97	**89**	66	2.44
13	87.79	**89**	81	1.16	85.56	**89**	75	1.94
14	88.20	**89**	81	0.99	85.94	**89**	75	1.71
15	88.45	**89**	81	0.81	86.13	**89**	72	1.58
16	88.66	**89**	83	0.65	86.39	**89**	79	1.31
17	88.80	**89**	84	0.48	86.44	**89**	80	1.19
18	88.88	**89**	86	0.35	**86.51**	**89**	80	1.16
19	88.93	**89**	86	0.27	86.48	**89**	82	1.19
20	88.96	**89**	86	0.21	86.46	**89**	81	1.17
21	88.97	**89**	87	0.19	86.41	**89**	82	**1.12**
22	88.98	**89**	87	0.15	86.34	**89**	82	1.17
23	88.95	**89**	87	0.14	86.37	**89**	82	1.15
24	**88.99**	**89**	88	0.11	86.25	**89**	82	1.20
25	**88.99**	**89**	88	0.12	86.26	**89**	82	1.22
26	**88.99**	**89**	88	0.10	86.16	**89**	80	1.28
27	**88.99**	**89**	88	0.11	86.13	**89**	81	1.26
28	**88.99**	**89**	88	0.10	86.04	**89**	79	1.35
29	**88.99**	**89**	88	**0.09**	86.07	**89**	81	1.33
30	**88.99**	**89**	88	0.10	86.00	**89**	79	1.37

Best value obtained by each method is in bold

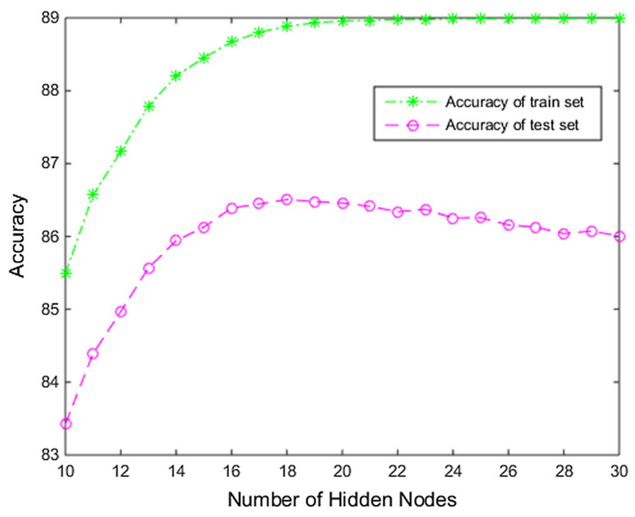

Fig. 4 Prediction of test samples (SaELM) with $N_{min} = 10$, $N_{max} = 30$ and $N_{Interval} = 1$ for the Italian wine classification problem

6.1.3 *Comparisons of SaELM with BP, ELM and GRNN*

In order to further evaluate the performance of SaELM, the SaELM is compared with the traditional FNN (BP and GRNN) and the basic ELM.

Table 3 Accuracy of BP, ELM, GRNN and SaELM for the Italian wine classification problem

	Training set				Test set			
	Mean	Best	Worst	Std	Mean	Best	Worst	Std
BP	87.08	**89**	52	2.67	84.12	**89**	43	3.80
ELM	**88.96**	**89**	**86**	**0.21**	86.46	**89**	**81**	1.17
GRNN	88.50	**89**	84	0.74	85.58	**89**	79	1.34
SaELM	88.88	**89**	**86**	0.35	**86.51**	**89**	80	**1.16**

Best value obtained by each method is in bold

For ELM, the number of neurons in the hidden layer is 20, and the activation function is S function. For BP NN, epochs = 50, learning rate = 0.1 and objective = 0.00004. For GRNN, cyclic training method is used in order to select the best SPREAD value, making GRNN achieve the best prediction.

As intelligent algorithms always have some randomness, each run will generate the different results. In order to get a typical statistical performance, 2000 implementations are conducted for each method. The results are given in Table 3.

In Table 3, for training set, the best performance and the average performance of BP, ELM, GRNN and SaELM have little difference. Moreover, ELM performs slightly better than BP, GRNN and SaELM. For test set, the average prediction accuracy of SaELM is better than BP, ELM and GRNN (86.51/89 = 97.20 %, 84.12/89 = 94.52 %, 86.46/89 = 97.15 % and 85.58/89 = 96.17 %). The results have proven that SaELM is suitable for classification and pattern recognition problems. Compared with the traditional FNN (BP and GRNN) and ELM, SaELM has a higher prediction accuracy.

6.2 Iris classification problem

In this section, the iris classification problem is solved by SaELM algorithm in order to investigate its performance for classification problem.

6.2.1 *Problem description*

Classification and identification of plant is an important basic work of production management and agroforestry in botany research and is of great significance for distinguishing plant species, exploring the genetic relationship between plants and elucidating the evolution rules for plant systems. At present, the plant species are identified by the classification retrieval table, but this method takes more time, and the establishment of the retrieval table is a time-consuming and laborious work, which needs to invest a lot of manpower and material resources.

The blade is an important part of plant leaves, and its outer contour is the main morphological characteristics. Based on the extraction of leaf shape features, computer-aided classification and recognition is becoming the main direction of study and is also the focus of the research hot spot.

A total of 150 sets of different iris (setosa, versicolor and virginica) are collected, which contains four kinds of property: sepal length, sepal width, petal length and petal width. Petal length, petal width and iris type have a good linear relationship, while sepal length, sepal width and iris types showed a nonlinear relationship. The source for iris data is also originated from UCI database (https://archive.ics.uci.edu/ml/datasets/Iris). Different kinds of iris are identified with 1, 2 and 3 (number 1–50 for setosa, 51–100 for versicolor, 101–150 for virginica). Each sample contains four dimensions. Each category is divided into two parts: training set, and test set.

6.2.2 *Performance of SaELM*

For iris classification problem, the parameters used are same with wine classification problem, as discussed in Sect. 6.1.2. The results are given in Table 4 and Fig. 5.

In Table 4, we can see that SaELM reaches the maximal accuracy 94.41 % (28.32/30) when $N_{\text{best}} = 25$. When N reaches 5–30, the SaELM has enough information learnt from training set to classify the test set. Therefore, the accuracy of training set and test set is becoming much better as the N increases. When N reaches certain point between 20 and 30, the SaELM approaches the best state, both training set and test set have the highest accuracy. After that, SaELM is in over-fitting state. Therefore, the accuracy of training set reaches the maximum accuracy and keeps almost unchanged all the time, and the test set is getting slower and slower as the N increases.

And then, $N_{\min}$, $N_{\max}$ and N_{Interval} are updated according to Eqs. (13)–(15), as shown in Eqs. (22)–(24).

$$\begin{aligned} N_{\min} &= \max(N_{\text{best}} - Q * N_{\text{Interval}},\ N_{\min}) \\ &= \max(25 - 2 \times 5,\ 10) \\ &= 15 \end{aligned} \tag{22}$$

$$\begin{aligned} N_{\max} &= \min(N_{\text{best}} + Q * N_{\text{Interval}},\ N_{\max}) \\ &= \min(25 + 2 \times 5,\ 120) \\ &= 35 \end{aligned} \tag{23}$$

$$\begin{aligned} N_{\text{Interval}} &= \text{floor}(N_{\text{Interval}}/L) \\ &= \text{floor}(5/4) \\ &= 1 \end{aligned} \tag{24}$$

Therefore, M is 21 according to Eq. (12). That is, for the next round, $N_1 = N_{\min} = 15$, $N_2 = N_{\min} + N_{\text{Interval}} = 16$, $N_3 = N_{\min} + 2 \times N_{\text{Interval}} = 17$, …, $N_j = N_{\min} + (j-1) \times N_{\text{Interval}} = 10 + (j-1) \times 1 = 14 + j$, …, $N_M = N_{21} = N_{\max} = 35$. The results are given in Table 5 and Fig. 6.

Table 4 Accuracy of SaELM with $N_{\min} = 5$, $N_{\max} = 120$ and $N_{\text{Interval}} = 5$ for the iris classification problem

N	Training set				Test set			
	Mean	Best	Worst	Std	Mean	Best	Worst	Std
5	106.42	119	80	7.81	25.78	30	16	2.58
10	116.74	119	101	2.19	27.98	30	24	1.11
15	118.58	119	110	0.98	28.16	30	26	0.76
20	118.86	119	116	0.61	28.29	**30**	**26**	**0.70**
25	118.96	119	117	0.35	**28.32**	30	**26**	0.75
30	118.91	119	118	0.16	28.27	30	**26**	0.81
35	118.95	**120**	118	0.19	27.97	30	26	0.88
40	119.13	**120**	118	0.29	27.62	30	25	0.99
45	119.42	**120**	117	0.36	27.20	30	24	1.01
50	119.41	**120**	118	0.43	26.79	30	24	1.01
55	119.69	**120**	118	0.51	26.34	29	23	1.00
60	119.89	**120**	118	0.46	26.19	29	23	1.01
65	119.98	**120**	119	0.31	26.16	29	22	1.06
70	119.99	**120**	119	0.11	26.21	29	22	1.21
75	119.99	**120**	119	0.03	26.11	29	20	1.44
80	**120**	**120**	119	0.02	25.69	29	19	1.65
85	**120**	**120**	**120**	**0**	24.83	29	18	1.85
90	**120**	**120**	**120**	**0**	23.63	29	15	2.11
95	**120**	**120**	**120**	**0**	22.38	28	15	2.20
100	**120**	**120**	**120**	**0**	20.97	27	11	2.24
105	**120**	**120**	**120**	**0**	19.39	28	11	2.43
110	**120**	**120**	**120**	**0**	17.65	25	8	2.58
115	**120**	**120**	**120**	**0**	15.14	24	5	2.90
120	**120**	**120**	**120**	**0**	14.34	25	5	2.87

Best value obtained by each method is in bold

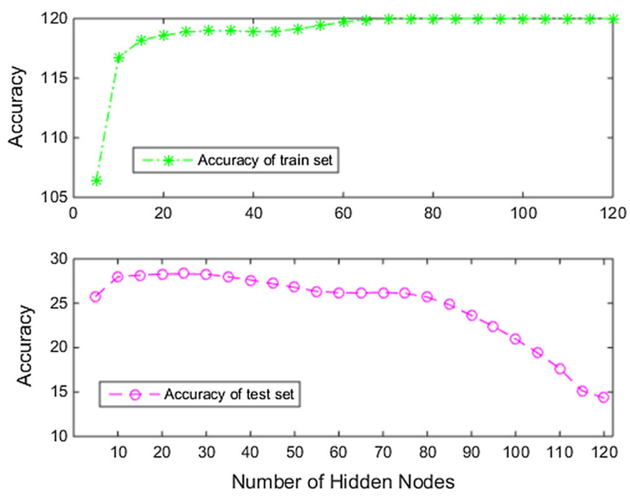

Fig. 5 Prediction of test samples (SaELM) with $N_{\min} = 5$, $N_{\max} = 120$ and $N_{\text{Interval}} = 5$ for the iris classification problem

Table 5 Accuracy of SaELM with $N_{\min} = 15$, $N_{\max} = 35$ and $N_{\text{Interval}} = 1$ for the iris classification problem

N	Training set				Test set			
	Mean	Best	Worst	Std	Mean	Best	Worst	Std
15	118.76	**120**	115	0.80	29.12	**30**	**26**	0.78
16	118.82	**120**	116	0.76	29.21	**30**	26	0.72
17	118.92	**120**	116	0.74	29.27	**30**	27	0.74
18	118.97	**120**	116	0.73	29.34	**30**	27	0.72
19	119.07	**120**	116	0.71	29.37	**30**	24	0.70
20	119.18	**120**	116	0.66	29.42	**30**	27	0.67
21	119.25	**120**	116	0.64	29.48	**30**	27	**0.64**
22	119.33	**120**	117	0.63	29.48	**30**	27	**0.64**
23	119.44	**120**	117	0.57	**29.67**	**30**	27	**0.64**
24	119.48	**120**	117	0.57	29.48	**30**	27	0.65
25	119.56	**120**	117	0.54	29.46	**30**	27	0.66
26	119.61	**120**	**118**	0.53	29.44	**30**	**28**	0.65
27	119.67	**120**	**118**	0.49	29.44	**30**	27	0.65
28	119.71	**120**	**118**	0.46	29.40	**30**	**28**	0.65
29	119.74	**120**	**118**	0.45	29.34	**30**	27	0.68
30	119.76	**120**	**118**	0.43	29.29	**30**	**28**	0.68
31	119.76	**120**	**118**	0.44	29.20	**30**	27	0.69
32	119.77	**120**	**118**	0.43	29.12	**30**	27	0.71
33	119.78	**120**	**118**	**0.41**	29.04	**30**	27	0.71
34	**119.79**	**120**	**118**	**0.41**	28.94	**30**	27	0.70
35	119.77	**120**	**118**	0.43	28.85	**30**	27	0.70

Best value obtained by each method is in bold

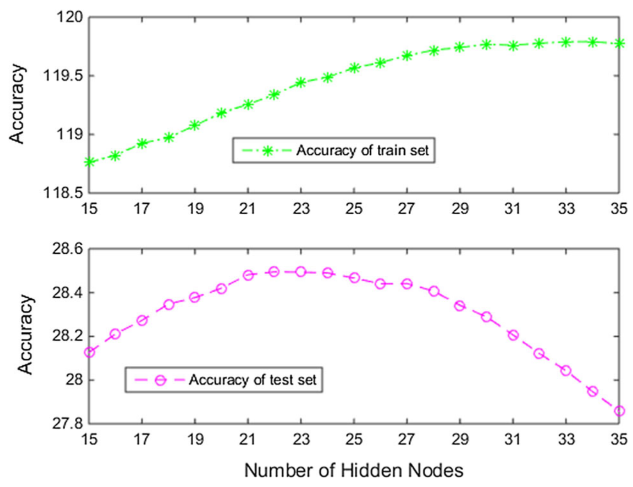

Fig. 6 Prediction of test samples (SaELM) with $N_{\min} = 15$, $N_{\max} = 35$ and $N_{\text{Interval}} = 1$ for the iris classification problem

In Table 5, we can see that SaELM reaches the maximal accuracy 98.92 % (29.67/30) when $N_{\text{best}} = 23$. When N reaches 15–22, the SaELM is under learn. Therefore, the accuracy of training set and test set is becoming much better as the N increases. SaELM has the similar accuracy when the number of hidden nodes N is between 21 and 24. As a matter of fact, as shown in Table 5, the SaELM is in the best state when N is 23, and the accuracy of training set and test set reaches climax. After that, SaELM is in over-fitting state, and the accuracy of training set is therefore the highest value and almost unchanged all the time. However, the test set is getting slower as the N increases. And then, $N_{\min}$, $N_{\max}$ and N_{Interval} are updated according to Eqs. (13)–(15), as shown in Eqs. (25)–(27).

$$\begin{aligned} N_{\min} &= \max(N_{\text{best}} - Q * N_{\text{Interval}}, N_{\min}) \\ &= \max(23 - 2 \times 1, 15) \\ &= 21 \end{aligned} \tag{25}$$

$$\begin{aligned} N_{\max} &= \min(N_{\text{best}} + Q * N_{\text{Interval}}, N_{\max}) \\ &= \min(23 + 2 \times 1,\ 35) \\ &= 25 \end{aligned} \tag{26}$$

$$\begin{aligned} N_{\text{Interval}} &= \text{floor}(N_{\text{Interval}}/L) \\ &= \text{floor}(1/4) \\ &= 0 \end{aligned} \tag{27}$$

According to Eq. (27), N_{Interval} is equal to zero; therefore, the SaELM stops and returns the best neuron number $N_{\text{best}} = 27$ and its responding accuracy 98.92 % (29.67/30).

6.2.3 Comparisons of SaELM with BP, ELM and GRNN

In order to further evaluate the performance of SaELM when solving the iris classification problem, the SaELM is compared with the traditional FNN (BP and GRNN) and the basic ELM.

For the parameters used in BP, ELM and GRNN, they are set as same as Sect. 6.1.3. In order to remove the influence of the randomness and get a typical statistical performance, 2000 implementations are conducted for each method (see Table 6).

Table 6 shows that, for training set, the best performance and the average performance of BP, ELM, GRNN and SaELM have little difference. Moreover, SaELM performs slightly better than BP, ELM and GRNN. For test set, the average prediction accuracy of SaELM is better than BP, ELM and GRNN (28.83/30 = 96.10 %, 29.11/30 = 97.03 %, 29.31/30 = 97.70 % and 29.67/30 = 98.91 %). The results have proven that SaELM is suitable for classification and pattern recognition problems. Compared with the traditional (BP and GRNN) and ELM, SaELM has a higher prediction accuracy.

According to the experimental results on Italian wine and iris classification problems, more hidden nodes require more computations, but not always lead to good results.

Table 6 Accuracy of BP, ELM, GRNN and SaELM for the iris classification problem

	Training set				Test set			
	Mean	Best	Worst	Std	Mean	Best	Worst	Std
BP	117.80	**120**	107	1.35	28.83	**30**	22	1.82
ELM	118.29	**120**	114	0.90	29.11	**30**	25	0.89
GRNN	114.18	119	111	1.27	29.31	**30**	24	1.17
SaELM	**119.67**	**120**	**118**	**0.49**	**29.67**	**30**	**27**	**0.65**

Best value obtained by each method is in bold

Table 7 Comparisons between SaELM and other methods at $\alpha = 0.05$ on a two-tailed t tests

	Training set	Test set
Wine		
BP	6.07E−46	2.91E−48
ELM	1.37E−67	2.26E−60
BP	8.31E−54	8.41E−57
Iris		
ELM	8.59E−8	1.32E−8
GRNN	3.59E−8	2.30E−8
GRNN	2.16E−19	0.077

Therefore, proper number of hidden nodes is the best choice for the SaELM.

6.3 Comparisons using t test and computational requirements

In this section, a kind of statistical analysis [56], t test, is used to further evaluate four methods as used in [57]. Based on the final results of 2000 independent trials on Italian wine and iris classification problem, Table 7 presents the t values on two classification problems of the two-tailed test with the 5 % level of significance between SaELM and BP, EML and GRNN. If the degrees of freedom (DoF) in t test is too big, the t test will become Gaussian distribution. Therefore, forty implementations are selected to perform t test. Accordingly, in the table, the value of t with 78 DoF is significant at $\alpha = 0.05$ by a two-tailed test. Table 7 shows that there is no significant difference between SaELM and other methods though SaELM indeed performs better than BP, ELM and GRNN, as proved in Sects. 6.1 and 6.2.

Computational requirements for four methods are collected as shown in Table 8. BP takes more time to train the networks than other three methods. For GRNN, cyclic training method is used in order to select the best SPREAD value in order to get the best prediction, and this selection process is a time-consuming work. Therefore, both of them take more time than the other two methods. ELM uses the least time when solving the Italian wine and iris classification problems. However, if the improper number of hidden nodes is selected, the ELM may lead to very bad accuracy. For SaELM, it can always find the best number of hidden nodes and generate the satisfactory classification accuracy.

Table 8 Computational comparisons between SaELM and other methods (s)

	Wine	Iris
BP	6.11	0.57
ELM	**0.71**	**0.12**
GRNN	7.50	0.17
SaELM	0.89	0.14

Best value obtained by each method is in bold

7 Conclusion

In order to overcome the disadvantages of SLFN, such as slow training speed, easy to fall into local minima and sensitive to learning rate, an improved algorithm for SLFN-ELM was proposed by Huang et al. [2]. In order to further overcome the sensitivity of neuron number in hidden layer, a SaELM is originally proposed. In this paper, a self-adaptive mechanism is combined with the basic ELM to form SaELM algorithm. SaELM can always select the best neuron number in hidden layer to construct the optimal networks. This method trains fast and can obtain the global optimal solution and has good generalization performance. And SaELM randomly generated the connection weights between input layer and hidden layer, and there is no adjustment in the training process. Therefore, there are no parameters to adjust in SaELM. Most importantly, a unique optimal solution can be always obtained. In order to prove the performance of the SaELM, it is used to solve Italian wine and iris classification problems. By comparing to other traditional methods (such as BP, ELM and GRNN), this method has a faster learning speed and better generalization performance.

In future, our research highlights would be focused on the following points. On the one hand, SaELM will be used to solve other regression and classification problems, and their results can be further compared to other methods. On the other hand, we will hybridize SaELM with some metaheuristic algorithms, so as to further improve the performance of SaELM.

Acknowledgments This work was supported by Research Fund for the Doctoral Program of Jiangsu Normal University (No. 13XLR041) and the National Natural Science Foundation of China (Nos. 61402207, 61100167, and 61272297), the Natural Science Foundation of Jiangsu Province, China, under Grant No. BK2011204, Qing Lan Project.

References

1. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501. doi:10.1016/j.neucom.2005.12.126
2. Huang G-B, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16–18):3056–3062. doi:10.1016/j.neucom.2007.02.009
3. Huang G-B, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cyber 2(2):107–122. doi:10.1007/s13042-011-0019-y
4. Horata P, Chiewchanwattana S, Sunat K (2013) Robust extreme learning machine. Neurocomputing 102:31–44. doi:10.1016/j.neucom.2011.12.045
5. 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(9):1498–1505. doi:10.1109/TNNLS.2012.2202289
6. Jun W, Shitong W, Chung F (2011) Positive and negative fuzzy rule system, extreme learning machine and image classification. Int J Mach Learn Cyber 2(4):261–271. doi:10.1007/s13042-011-0024-1
7. Pouzols FM, Lendasse A (2010) Evolving fuzzy optimally pruned extreme learning machine for regression problems. Evol Syst 1(1):43–58. doi:10.1007/s12530-010-9005-y
8. Li G, Niu P (2011) An enhanced extreme learning machine based on ridge regression for regression. Neural Comput Appl 22(3–4):803–810. doi:10.1007/s00521-011-0771-7
9. Zhai J-H, Xu H-Y, Wang X-Z (2012) Dynamic ensemble extreme learning machine based on sample entropy. Soft Comput 16(9):1493–1502. doi:10.1007/s00500-012-0824-6
10. Zong W, Huang G-B, Chen Y (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242. doi:10.1016/j.neucom.2012.08.010
11. He Q, Shang T, Zhuang F, Shi Z (2013) Parallel extreme learning machine for regression based on MapReduce. Neurocomputing 102:52–58. doi:10.1016/j.neucom.2012.01.040
12. Wang D, Alhamdoosh M (2013) Evolutionary extreme learning machine ensembles with size control. Neurocomputing 102:98–110. doi:10.1016/j.neucom.2011.12.046
13. Feng G, Huang G-B, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357. doi:10.1109/TNN.2009.2024147
14. Wang L, Huang Y, Luo X, Wang Z, Luo S (2011) Image deblurring with filters learned by extreme learning machine. Neurocomputing 74(16):2464–2474. doi:10.1016/j.neucom.2010.12.035
15. Iosifidis A, Tefas A, Pitas I (2013) Dynamic action recognition based on dynemes and extreme learning machine. Pattern Recogn Lett 34(15):1890–1898. doi:10.1016/j.patrec.2012.10.019
16. Song Y, Crowcroft J, Zhang J (2012) Automatic epileptic seizure detection in EEGs based on optimized sample entropy and extreme learning machine. J Neurosci Methods 210(2):132–146. doi:10.1016/j.jneumeth.2012.07.003
17. Chacko BP, Vimal Krishnan VR, Raju G, Babu Anto P (2011) Handwritten character recognition using wavelet energy and extreme learning machine. Int J Mach Learn Cyber 3(2):149–161. doi:10.1007/s13042-011-0049-5
18. Zheng W, Qian Y, Lu H (2012) Text categorization based on regularization extreme learning machine. Neural Comput Appl 22(3–4):447–456. doi:10.1007/s00521-011-0808-y
19. Hu X-F, Zhao Z, Wang S, Wang F-L, He D-K, Wu S-K (2007) Multi-stage extreme learning machine for fault diagnosis on hydraulic tube tester. Neural Comput Appl 17(4):399–403. doi:10.1007/s00521-007-0139-1
20. Suresh S, Venkatesh Babu R, Kim HJ (2009) No-reference image quality assessment using modified extreme learning machine classifier. Appl Soft Compt 9(2):541–552. doi:10.1016/j.asoc.2008.07.005
21. Chen FL, Ou TY (2011) Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry. Expert Syst Appl 38(3):1336–1345. doi:10.1016/j.eswa.2010.07.014
22. Malathi V, Marimuthu NS, Baskar S, Ramar K (2011) Application of extreme learning machine for series compensated transmission line protection. Eng Appl Artif Intel 24(5):880–887. doi:10.1016/j.engappai.2011.03.003
23. Zhao X (2010) A perturbed particle swarm algorithm for numerical optimization. Appl Soft Compt 10(1):119–124. doi:10.1016/j.asoc.2009.06.010
24. Zhao X, Liu Z, Yang X (2014) A multi-swarm cooperative multistage perturbation guiding particle swarm optimizer. Appl Soft Compt 22:77–93. doi:10.1016/j.asoc.2014.04.042
25. Li X, Yin M (2012) Application of differential evolution algorithm on self-potential data. PLoS One 7(12):e51199. doi:10.1371/journal.pone.0051199
26. Zou D, Liu H, Gao L, Li S (2011) An improved differential evolution algorithm for the task assignment problem. Eng Appl Artif Intel 24(4):616–624. doi:10.1016/j.engappai.2010.12.002
27. Li X, Yin M (2014) Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm. Nonlinear Dynam 77(1–2):61–71. doi:10.1007/s11071-014-1273-9
28. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38(10):1759–1763. doi:10.1016/j.patcog.2005.03.028
29. Cao J, Lin Z, Huang G-B (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36(3):285–305. doi:10.1007/s11063-012-9236-y
30. Gandomi AH, Yang X-S, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35. doi:10.1007/s00366-011-0241-y
31. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Abraham A, Carvalho A, Herrera F, Pai V (eds) Proceeding of world congress on nature & biologically inspired computing (NaBIC 2009), Coimbatore, India, December 2009. IEEE Publications, USA, pp 210–214
32. Wang G-G, Guo L, Duan H, Wang H (2014) A new improved firefly algorithm for global numerical optimization. J Comput Theor Nanosci 11(2):477–485. doi:10.1166/jctn.2014.3383
33. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84. doi:10.1504/IJBIC.2010.032124
34. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. doi:10.1016/j.advengsoft.2013.12.007
35. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68. doi:10.1177/003754970107600201
36. Wang G, Guo L, Wang H, Duan H, Liu L, Li J (2014) Incorporating mutation scheme into krill herd algorithm for global numerical optimization. Neural Comput Appl 24(3–4):853–871. doi:10.1007/s00521-012-1304-8
37. Zou D, Gao L, Li S, Wu J (2011) Solving 0–1 knapsack problem by a novel global harmony search algorithm. Appl Soft Compt 11(2):1556–1564. doi:10.1016/j.asoc.2010.07.019
38. Wang G, Guo L, Duan H, Wang H, Liu L, Shao M (2013) Hybridizing harmony search with biogeography based optimization for global numerical optimization. J Comput Theor Nanosci 10(10):2318–2328. doi:10.1166/jctn.2013.3207

39. Simon D (2008) Biogeography-based optimization. IEEE Trans Evolut Comput 12(6):702–713. doi:10.1109/TEVC.2008.919004
40. Wang G-G, Gandomi AH, Alavi AH (2014) An effective krill herd algorithm with migration operator in biogeography-based optimization. Appl Math Model 38(9–10):2454–2462. doi:10.1016/j.apm.2013.10.052
41. Saremi S, Mirjalili S, Lewis A (2014) Biogeography-based optimisation with chaos. Neural Comput Appl 25(5):1077–1097. doi:10.1007/s00521-014-1597-x
42. Li X, Yin M (2013) Multiobjective binary biogeography based optimization for feature selection using gene expression data. IEEE Trans Nanobiosci 12(4):343–353. doi:10.1109/TNB.2013.2294716
43. Li X, Zhang J, Yin M (2014) Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput Appl 24(7–8):1867–1877. doi:10.1007/s00521-013-1433-8
44. Mirjalili S, Wang G-G, Coelho LdS (2014) Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput Appl 25(6):1423–1435. doi:10.1007/s00521-014-1629-6
45. Mirjalili S, Lewis A (2014) Adaptive gbest-guided gravitational search algorithm. Neural Comput Appl 25(7–8):1569–1584. doi:10.1007/s00521-014-1640-y
46. Mirjalili S, Mohd Hashim SZ, Moradian Sardroudi H (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137. doi:10.1016/j.amc.2012.04.069
47. Zhang Z, Zhang N, Feng Z (2014) Multi-satellite control resource scheduling based on ant colony optimization. Expert Syst Appl 41(6):2816–2823. doi:10.1016/j.eswa.2013.10.014
48. Zhang Z, Feng Z (2012) Two-stage updating pheromone for invariant ant colony optimization algorithm. Expert Syst Appl 39(1):706–712. doi:10.1016/j.eswa.2011.07.062
49. Gandomi AH (2014) Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans 53(4):1168–1183. doi:10.1016/j.isatra.2014.03.018
50. Gandomi AH, Yang X-S, Alavi AH, Talatahari S (2013) Bat algorithm for constrained optimization tasks. Neural Comput Appl 22(6):1239–1255. doi:10.1007/s00521-012-1028-9
51. Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5):464–483. doi:10.1108/02644401211235834
52. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845. doi:10.1016/j.cnsns.2012.05.010
53. Wang G-G, Gandomi AH, Alavi AH (2014) Stud krill herd algorithm. Neurocomputing 128:363–370. doi:10.1016/j.neucom.2013.08.031
54. Guo L, Wang G-G, Gandomi AH, Alavi AH, Duan H (2014) A new improved krill herd algorithm for global numerical optimization. Neurocomputing 138:392–402. doi:10.1016/j.neucom.2014.01.023
55. Wang G-G, Gandomi AH, Alavi AH (2013) A chaotic particle-swarm krill herd algorithm for global numerical optimization. Kybernetes 42(6):962–978. doi:10.1108/K-11-2012-0108
56. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18. doi:10.1016/j.swevo.2011.02.002
57. Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H (2014) Chaotic krill herd algorithm. Inf Sci 274:17–34. doi:10.1016/j.ins.2014.02.123