Neurocomputing xx (xxxx) xxxx–xxxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An improved incremental constructive single-hidden-layer feedforward networks for extreme learning machine based on particle swarm optimization

Fei Han[a,*], Min-Ru Zhao[a], Jian-Ming Zhang[a], Qing-Hua Ling[a,b]

[a] School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, Jiangsu, China
[b] School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, China

ARTICLE INFO

Keywords:
Extreme learning machine
Particle swarm optimization
Network structure
Generalization performance
Condition value

ABSTRACT

How to determine the network structure is an open problem in extreme learning machine (ELM). Error minimized extreme learning machine (EM-ELM) is a simple and efficient approach to determine the number of hidden nodes. However, similar to other constructive ELM, EM-ELM lays much emphasis on the convergence accuracy, which may obtain a single-hidden-layer feedforward neural networks (SLFN) with good convergence performance but bad condition. In this paper, an effective approach based on error minimized ELM and particle swarm optimization (PSO) is proposed to adaptively determine the structure of SLFN for regression problem. In the new method, to establish a compact and well-conditioning SLFN, the hidden node optimized by PSO is added to the SLFN one by one. Moreover, not only the regression accuracy but also the condition value of the hidden output matrix of the network is considered in the optimization process. Experiment results on various regression problems verify that the proposed algorithm achieves better generalization performance with fewer hidden nodes than other constructive ELM.

1. Introduction

Single-hidden-layer feedforward neural networks (SLFN) has been proved to be universal approximator and are widely used for regression and classification problems [1]. The learning speed of SLFN is in general far slower than required because gradient-based learning algorithms are extensively used to train neural networks, which all parameters of the networks are tuned iteratively [2–4]. To overcome the defects of gradient-based learning algorithms, random vector functional link networks (RVFL) was proposed [5] where actual values of the weights from the input layer to hidden layer can be randomly generated in a suitable domain and kept fixed in the learning stage. As an effective learning algorithm for RVFL, extreme learning machine (ELM) was proposed by Huang et al. [6], which randomly chose input weights and hidden biases and analytically determined the output weights. Without iteratively tuning the parameters, ELM could converge much faster with better generalization performance than gradient-descent learning algorithms [6–8].

Because of randomly choosing the input weights and the hidden biases, the traditional ELM inevitably needs more hidden nodes and the network structure becomes more complexity, which may further affect the generalization performance of the network. In some ELM [6,9–11], the size of the network size is predetermined by trial and error before training. Using these methods to determine the size of the network is not only time consuming but also ineffective in some cases. A class of constructive ELM [12–17] were proposed to automatically design the network architecture. In incremental ELM (I-ELM) [12], the hidden nodes were added one by one and the output weights of the existing network were frozen when a new hidden node was added. During the learning procedure in I-ELM, as more and more hidden nodes including some nodes playing a very minor role in the network output were added, the network complexity may be increased. An enhanced I-ELM (EI-ELM) [14] was proposed to obtain more compact network architecture. In EI-ELM, several hidden nodes were randomly generated, and the one leading to the minimum residual error would be added to the existing network. Feng et al. [15] proposed an error minimized ELM (EM-ELM), which added random hidden nodes with varying size to the existing network. Moreover, the output weights updated incrementally during the network growth, which significantly reduced the computational complexity. Zhang et al. [16] proposed an ELM with adaptive growth of hidden nodes (AG-ELM). In AG-ELM, the network size was determined by an adaptive way that the existing network may be replaced by newly generated networks which had fewer hidden nodes and better generalization performance. However,

* Corresponding author.

http://dx.doi.org/10.1016/j.neucom.2016.09.092
Received 1 January 2016; Received in revised form 4 September 2016; Accepted 19 September 2016
Available online xxxx
0925-2312/

in AG-ELM both the hidden nodes and the output weights were randomly generated. Based on EM-ELM and AG-ELM, Zhang et al. [17] proposed dynamic ELM (D-ELM) where the hidden nodes can be recruited dynamically according to their significance to network performance. In D-ELM, not only the parameters can be adjusted but also the architecture can be self-adapted simultaneously.

In the implementation of the above constructive ELM, more efforts have been emphasized on the convergence accuracy of the ELM, whereas the numerical stability of the constructed SLFN is generally ignored [18,19]. Numerical stability is an important aspect of a system. An ill-conditioned system may have its solutions very sensitive to perturbation in data. The conditioning of a SLFN in ELM can be quantitatively characterized by the condition value of the hidden output matrix which is determined by input weights and input data [19,20]. Therefore, it is apparent that random input weights consider no robustness for the solution, and only an elaborate selection of input weights with respect to a particular set of input data may achieve a well-conditioned system.

Similar to genetic algorithm (GA) [21], particle swarm optimization (PSO) is also an effective global search technique, because it has good search ability, fast convergence and no complicated evolutionary operators [22–24]. Some algorithms such as PSO-ELM [24], IPSO-ELM [20] verified the effectiveness to use PSO to optimize the parameters of ELM. In PSO-ELM, boundary conditions were considered in PSO to optimize the input weights in ELM. In IPSO-ELM, the improved PSO optimized the input weights and hidden biases according to not only RMSE on validation set but also the norm of the output weights. However, the network structures in these PSO based ELM were predetermined by trial and error.

The classical constructive ELM have paid much attention to the convergence accuracy and structure without considering the condition of SLFN, and the condition of the SLFN is direct relevant to the input weights and hidden biases. Therefore, establishing an effective SLFN with selecting optimal input weights and hidden biases by PSO is a reasonable choice. In [25], we proposed a new method combining incremental EM-ELM and PSO called IPSO-EM-ELM to automatically determine the structure of SLFN for regression problem. To extend the work in [25], more details of IPSO-EM-ELM, additional experiments and discussion are added in this paper. In the IPSO-EM-ELM, hidden node is added one by one to the existing network, and PSO is used to optimize the input weights and hidden biases in the new hidden nodes. In the optimization process, not only the root mean squared error (RMSE) of training data but also the condition value of the hidden output matrix of the SLFN is considered to select the optimal hidden nodes, which may establish more compact and well-conditioned SLFN with better generalization performance. The output weights are updated incrementally as the EM-ELM [15]. Finally, the experiment results on seven data from UCI database verify the effectiveness and efficiency of the proposed ELM.

The remainder of paper is organized as follows. Section 2 introduces some preliminaries of EM-ELM and PSO. The proposed method is presented in Section 3. Experiment results and discussion are presented in Section 4. Finally, the concluding remarks are given in Section 5.

2. Preliminaries

2.1. Error minimized extreme learning machine (EM-ELM)

Assume that an SLFN with d inputs, L hidden nodes, and one linear output can be represented in a general form with the following equation

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (1)$$

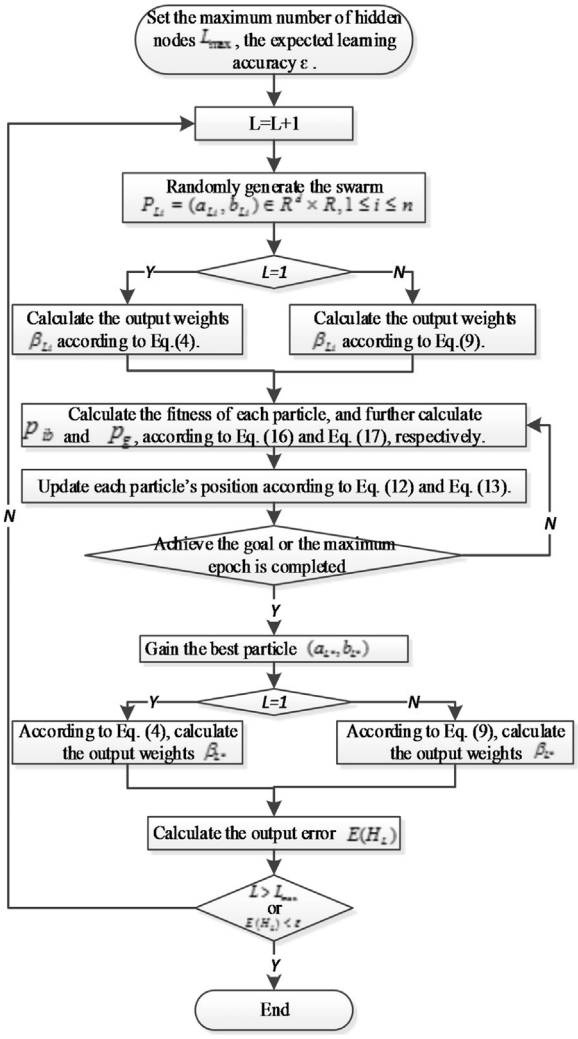


Fig. 1. The flowchart of the IPSO-EM-ELM.

Table 1
Specification of seven benchmark data sets.

Datasets	Attributes	Cases	Training data	Testing data
Abalone	8	4177	2000	2177
Boston housing	13	506	250	256
California housing	8	20640	8000	12640
Census (House8L)	8	22784	10000	12784
Delta ailerons	5	7129	3000	4129
Delta elevators	6	9517	4000	5317
Machine CPU	6	209	100	109

where $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output of the ith hidden node with the randomly generated learning parameters $(\mathbf{a}_i, b_i) \in R^d \times R$. The parameters $\mathbf{a}_i$ and b_i are the input weights and hidden biases for the ith hidden node, respectively. $\beta_i \in R$ is the weight connecting the ith hidden node to the output node.

Given a set of training data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset R^d \times R$, the output of the network equaling to the target means

$$f_L(\mathbf{x}_j) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = t_j, j = 1, \ldots, N. \quad (2)$$

which can be equivalently expressed in the matrix form

$$\mathbf{H}\beta = \mathbf{T} \quad (3)$$

where

Table 2
Mean regression performance of the five ELM on the seven benchmark data.

Algorithm	Data Stop RMSE (ε)	Abalone 0.09	Boston housing 0.12	Calofornia Housing 0.16	Census (House8L) 0.09	Delta ailerons 0.05	Delta elevators 0.06	Machine CPU 0.08
I-ELM	Testing RMSE ± Dev	0.0988 ± 0.0020	0.1450 ± 0.0055	0.1680 ± 0.0055	0.0914 ± 0.0021	0.0538 ± 0.0056	0.0642 ± 0.0053	0.0616 ± 0.0074
	Training Times(s)	0.0546	0.0125	0.1538	0.716	0.0546	0.0850	0.0023
EI-ELM	Testing RMSE ± Dev	0.0973 ± 0.0006	0.1451 ± 0.0049	0.1606 ± 0.0004	0.0875 ± 0.0002	0.0513 ± 0.0012	0.0602 ± 0.0004	0.0570 ± 0.0112
	Training Times(s)	0.0499	0.0125	0.3752	0.4664	0.1084	0.1771	0.0039
EM-ELM	Testing RMSE ± Dev	0.0938 ± 0.0033	0.1502 ± 0.0119	0.1572 ± 0.0037	0.0859 ± 0.0020	0.0465 ± 0.0036	0.0575 ± 0.0023	0.0595 ± 0.0114
	Training Times(s)	0.0218	0.0016	0.0109	0.0086	0.0023	0.0008	0.0008
D-ELM	Testing RMSE ± Dev	0.0940 ± 0.0031	0.1492 ± 0.0095	0.1556 ± 0.0032	0.0864 ± 0.0018	0.0468 ± 0.0023	0.0575 ± 0.0020	0.0647 ± 0.0114
	Training Times(s)	0.0156	0.0585	0.0507	0.0874	0.0133	0.0281	0.0070
IPSO-EM-ELM	Testing RMSE ± Dev	**0.0919** ± 0.0037	**0.1444** ± 0.0099	**0.1549** ± 0.0025	**0.0857** ± 0.0022	**0.0432** ± 0.0019	**0.0572** ± 0.0013	**0.0575** ± 0.0091
	Training Times(s)	0.6162	0.3549	1.4095	3.1465	0.4672	0.6380	0.2457

$$\mathbf{H} = \begin{pmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ G(\mathbf{a}_1, b_1, \mathbf{x}_2) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_2) \\ \vdots & \vdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{pmatrix}_{N\times L}, \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_L \end{pmatrix}_{L\times 1} \mathbf{T} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}_{N\times 1}$$

$\boldsymbol{H}$ is the hidden-layer output matrix of the SLFN. With the input weights and hidden biases are randomly generated and the training data, the matrix $\boldsymbol{H}$ is determined. Thus, training SLFNs simply amounts to getting the solution of the linear system of the output weights vector β, which can be calculated as follows:

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

where $\mathbf{H}^{+}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

In EM-ELM, given target error$\varepsilon > 0$, an SLFN $f_{n_0}(\mathbf{x}) = \sum_{i=1}^{n_0} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})$ with n_0 hidden nodes is first initialized. $\mathbf{H}_1$ denotes the hidden-layer output matrix of this network. Then, the output weight matrix can be calculated by $\beta_1 = \mathbf{H}_1{}^{+}\mathbf{T}$.

If the network output error $E(\mathbf{H}_1) = \|\mathbf{H}_1\beta_1 - \mathbf{T}\| > \varepsilon$, new hidden nodes $\delta n_0 = n_1 - n_0$ will be added to the existing SLFN, and the new hidden-layer output matrix becomes$\mathbf{H}_2 = [\mathbf{H}_1, \delta\mathbf{H}_1]$. As proved in [12], the new network output error becomes smaller, that is$E(\mathbf{H}_2) = \|\mathbf{H}_2\beta_2 - \mathbf{T}\| \le E(\mathbf{H}_1) = \|\mathbf{H}_1\beta_1 - \mathbf{T}\|$, since β_2 is the least square solution of $\min\|\mathbf{H}_2\beta - \mathbf{T}\|$. Different from ELM where $\beta_2 = \mathbf{H}_2{}^{+}\mathbf{T}$ is obtained by calculating the Moore-Penrose inverse directly, EM-ELM proposes a fast output weights updating method to calculate $\mathbf{H}_2{}^{+}$as follows:

$$\mathbf{H}_2{}^{+} = (\mathbf{H}_2^T\mathbf{H}_2)^{-1}\mathbf{H}_2{}^T = \left(\begin{bmatrix}\mathbf{H}_1{}^T \\ \delta\mathbf{H}_1{}^T\end{bmatrix}[\mathbf{H}_1 \quad \delta\mathbf{H}_1]\right)^{-1}\begin{bmatrix}\mathbf{H}_1{}^T \\ \delta\mathbf{H}_1{}^T\end{bmatrix}. \tag{5}$$

Assume that

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \left(\begin{bmatrix}\mathbf{H}_1{}^T \\ \delta\mathbf{H}_1{}^T\end{bmatrix}[\mathbf{H}_1 \quad \delta\mathbf{H}_1]\right)^{-1} \tag{6}$$

According to the Schur complement, $\mathbf{H}_2{}^{+}$ can be deduced by the following equation:

$$\mathbf{H}_2{}^{+} = \begin{bmatrix} A_{11}\mathbf{H}_1{}^T + A_{12}\delta\mathbf{H}_1{}^T \\ A_{21}\mathbf{H}_1{}^T + A_{22}\delta\mathbf{H}_1{}^T \end{bmatrix} = \begin{bmatrix}\mathbf{U}_1 \\ \mathbf{D}_1\end{bmatrix} \tag{7}$$

where$\mathbf{D}_1 = ((\mathbf{I} - \mathbf{H}_1\mathbf{H}_1{}^{+})\delta\mathbf{H}_1{}^{+})^{+}$, $\mathbf{U}_1 = \mathbf{H}_1{}^{+}(\mathbf{I} - \delta\mathbf{H}_1\mathbf{D}_1)$. Thus, β_2 is calculated as follows:

$$\beta_2 = \mathbf{H}_2{}^{+}\mathbf{T} = \begin{bmatrix}\mathbf{U}_1 \\ \mathbf{D}_1\end{bmatrix}\mathbf{T} \tag{8}$$

Similarly, the output weights are updated incrementally as follows:

$$\mathbf{D}_k = ((\mathbf{I} - \mathbf{H}_k\mathbf{H}_k{}^{+})\delta\mathbf{H}_k{}^{+})^{+}, \cdot\mathbf{U}_k = \mathbf{H}_k{}^{+}(\mathbf{I} - \delta\mathbf{H}_k\mathbf{D}_k),$$

$$\beta_{k+1} = \mathbf{H}_{k+1}{}^{+}\mathbf{T} = \begin{bmatrix}\mathbf{U}_k \\ \mathbf{D}_k\end{bmatrix}\mathbf{T}, \tag{9}$$

where

$$\delta H_k = \begin{bmatrix} G(\mathbf{a}_{L_{k-1}+1}, b_{L_{k-1}+1}, \mathbf{X}_1) & \cdots & G(\mathbf{a}_{L_k}, b_{L_k}, \mathbf{X}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_{L_{k-1}+1}, b_{L_{k-1}+1}, \mathbf{X}_N) & \cdots & G(\mathbf{a}_{L_k}, b_{L_k}, \mathbf{X}_N) \end{bmatrix}_{N\times\delta L_{k-1}}.$$

EM-ELM allows that the number of the hidden-nodes added to the network can be different. As a special case, the hidden nodes can be added to the existing network one by one, that is, $\delta L_0 = \delta L_1 = \delta L_2 = \cdots = \delta L_k = \cdots = 1$. Then $\delta\mathbf{H}_k$ degenerates into a vector $\delta h_k = [G(\mathbf{a}_{L_{k-1}+1}, b_{L_{k-1}+1}, \mathbf{X}_1), \ \cdots \ G(\mathbf{a}_{L_k}, b_{L_k}, \mathbf{X}_N)]^T$ and the iterative formula becomes

$$\mathbf{D}_k = \frac{\delta h_k{}^T(\mathbf{I} - \mathbf{H}_k\mathbf{H}_k{}^{+})}{\delta h_k{}^T(\mathbf{I} - \mathbf{H}_k\mathbf{H}_k{}^{+})\delta h_k}, \cdot\mathbf{U}_k = \mathbf{H}_k{}^{+}(\mathbf{I} - \delta h_k\mathbf{D}_k) \tag{10}$$

Table 3
The mean number of hidden nodes, condition value of the hidden ouput matrix and norm value of the output weights in the five ELM on the seven benchmark data.

Algorithm	Data	Abalone	Boston housing	CaloforniaHousing	Census (House8L)	Delta ailerons	Delta elevators	Machine CPU
	Stop RMSE (ε)	0.09	0.12	0.16	0.09	0.05	0.06	0.08
I-ELM	Nodes ± Dev	137.35 ± 62.70	112.50 ± 46.36	199.60 ± 1.78	200 ± 0	157.30 ± 46.97	193.55 ± 15.26	30.60 ± 15.65
	K(H)	5.78e+5	1.33e+4	5.50e+5	2.62e+5	4.79e+6	4.50e+5	1.62e+5
	Norm value	0.7254	1.0664	1.6118	0.3697	1.0449	1.0992	0.7770
EI-ELM	Nodes ± Dev	42.75 ± 39.48	29.90 ± 8.45	81.65 ± 25.25	91.95 ± 16.98	42.65 ± 20.57	58.40 ± 16.39	7.80 ± 3.53
	K(H)	3.83e+4	386.75	5.43e+4	2.91e+4	5.35e+4	1.44e+4	318.68
	Norm value	0.6619	0.9067	1.5024	0.4167	0.8859	0.9540	0.8373
EM-ELM	Nodes ± Dev	4.85 ± 1.26	10.00 ± 2.10	6.95 ± 1.60	8.15 ± 1.66	4.25 ± 1.06	6.15 ± 1.56	4.50 ± 1.43
	K(H)	75.39	58.85	189.08	183.96	50.81	107.37	54.23
	Norm value	1.9897	2.1678	5.5184	3.1977	1.2621	1.8437	2.4067
D-ELM	Nodes ± Dev	3.55 ± 0.82	6.90 ± 1.44	5.30 ± 0.92	5.85 ± 1.30	3.45 ± 0.94	4.90 ± 0.85	3.40 ± 0.94
	K(H)	35.4669	30.0277	73.4690	79.7715	42.1727	37.9730	26.8442
	Norm value	1.3236	1.4909	4.2154	1.8242	0.9992	1.1635	1.7888
IPSO-EM-ELM	Nodes ± Dev	**2.85** ± 0.98	**3.25** ± 0.78	**3.10** ± 1.07	**3.90** ± 1.44	**1.95** ± 0.22	**2.50** ± 0.76	**2.55** ± 0.82
	K(H)	15.6726	6.9427	15.5599	19.0004	6.3696	9.4279	15.9201
	Norm value	1.0934	0.9936	2.5554	1.0508	0.8711	1.1233	1.8524

2.2. Particle swarm optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Eberhart and Kennedy [21,26]. PSO works by initializing a flock of birds randomly over the searching space, where each bird is called a particle with no quality or volume. Each particle flies with a certain velocity according to its momentum and the influence of its own previous best position (P_{ib}) as well as the best position of all particles (P_g). Assume that the dimension of searching space is D and the total number of particles is n. Then the original PSO is described as follows:

$$v_{id}(t+1) = v_{id}(t) + c_1 \times rand() \times [p_{ibd}(t) - x_{id}(t)] + c_2 \times rand() \times [p_{gd}(t) - x_{id}(t)] \tag{11}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad 1 \le i \le n, \quad 1 \le d \le D \tag{12}$$

where $v_i(t)$ and $x_i(t)$ denote the velocity vector and the position of the ith particle, respectively, at the tth iteration; $p_{ib}(t)$and $p_g(t)$denote the best position of the ith particle and the best position of all particle, respectively; c_1 and c_2 are the positive acceleration constants; $rand()$ is a random number between 0 and 1. In addition, it needs to place a limit on the velocity.

To improve the convergence performance of the original PSO, a modified particle swarm optimization [27] was proposed. An inertial weight was introduced in the velocity vector evolution equation described as follows:

$$v_{id}(t+1) = w(t) \times v_{id}(t) + c_1 \times rand() \times [p_{ibd}(t) - x_{id}(t)] + c_2 \times rand() \times [p_{gd}(t) - x_{id}(t)] \tag{13}$$

where w is the inertial weight. Shi and Eberhart [27] advised the linearly decreasing method to adjust the weight as follows:

$$w(t) = w_{ini} - \frac{(w_{ini} - w_{end})}{T_{\max}} \times t. \tag{14}$$

where t is the current iteration number; w_{ini}, w_{end}, $T_{\max}$ are the initial inertial weight, the final inertial weight and the maximum number of iteration, respectively.

3. The proposed method

How to determine the number of hidden nodes is a crucial issue in ELM. And it is important to build a compact and well-conditioned SLFN for improving the performance of ELM. Although the EM-ELM is an efficient algorithm to create a compact network, it lays much emphasis on the accuracy of solutions without considering the conditioning of the constructed SLFN and the hidden nodes are also randomly generated so that the performance of the EM-ELM is limited. In this study, PSO is used to optimize the randomly generated nodes. In addition to considering the convergence accuracy of the constructed SLFN, the condition value of the hidden output matrix is also considered in the optimization process to ensure that the constructed SLFN is well-conditioned. Whenever a new hidden node is added, the output weights will be incrementally updated to reduce computational cost as EM-ELM. The proposed method is referred to as IPSO-EM-ELM. The flowchart of the IPSO-EM-ELM is depicted in Fig. 1, and the detailed steps are presented as follows.

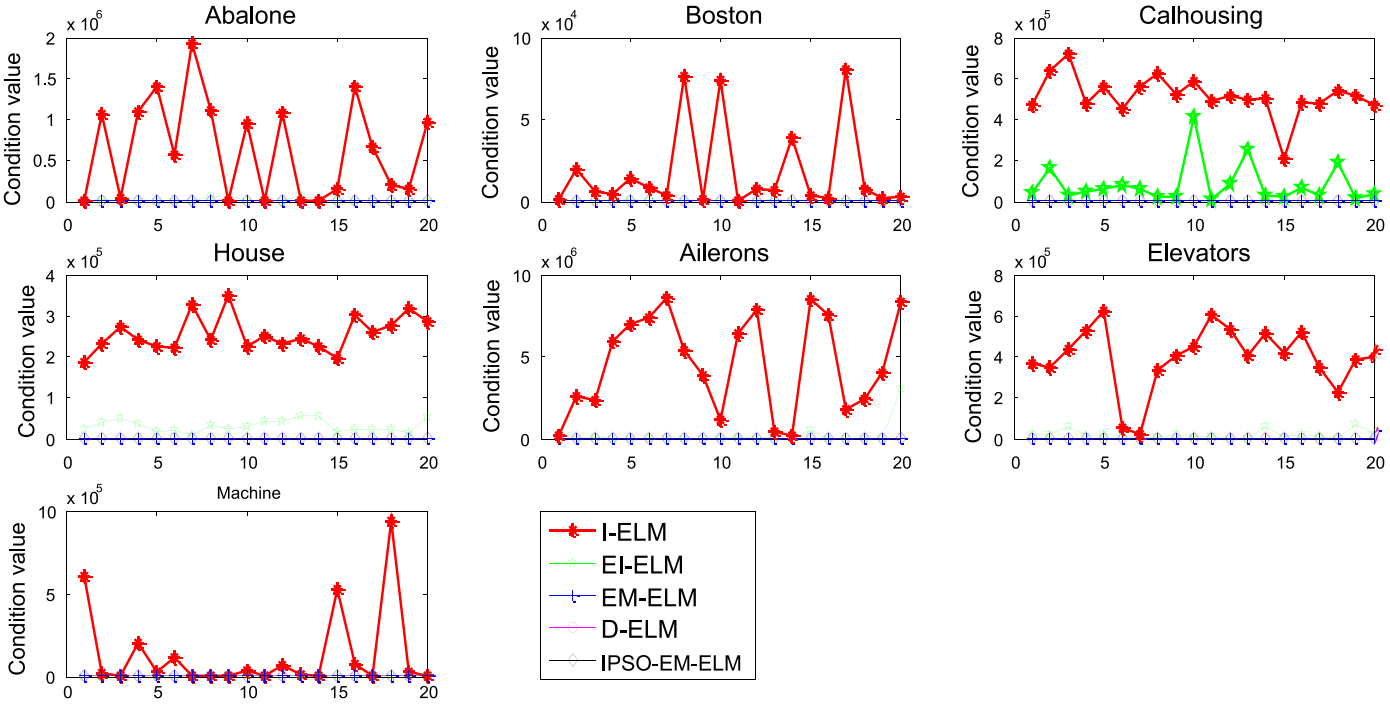
Given a set of training data $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N} \subset R^d \times R$, the maximum number of the hidden nodes $L_{\max}$, the expected training accuracy ε, and $L=0$:

- Step 1: Increase the number of hidden nodes $L=L+1$;
- Step 2: Use PSO to select the new hidden node:
- Substep 2.1: Randomly generate the swarm within the range of [−1,1]. Each particle is composed of input weights connected the new added hidden node to the input nodes and the hidden biases for the new added hidden node: $P_{Li} = (\mathbf{a}_{Li}, b_{Li})$.
- Substep 2.2: According Eq. (4) to calculate the output weightsβ_{Li}, $(1 \le i \le n, L = 1)$as $L=1$. According Eq. (9) to calculate the output weights β_{Li}, $(1 \le i \le n, L > 1)$as $L>1$. The fitness of each particle is calculated by Eq. (15):

$$f() = \|\mathbf{H}_{Li}\beta_{Li} - \mathbf{T}\| \tag{15}$$

Substep 2.3: With the fitness of all particles, the p_{ib} of each particle and the p_g of the swarm are computed by Eqs. (16) and (17), respectively:

$$p_{ib} = \begin{cases} p_i & (f(p_{ib}) - f(p_i) > \eta f(p_{ib})) \ or \\ & (f(p_{ib}) - f(p_i) < \eta f(p_{ib})) \\ & and\ (K_i < K_{ib})) \\ p_{ib} & else \end{cases} \tag{16}$$



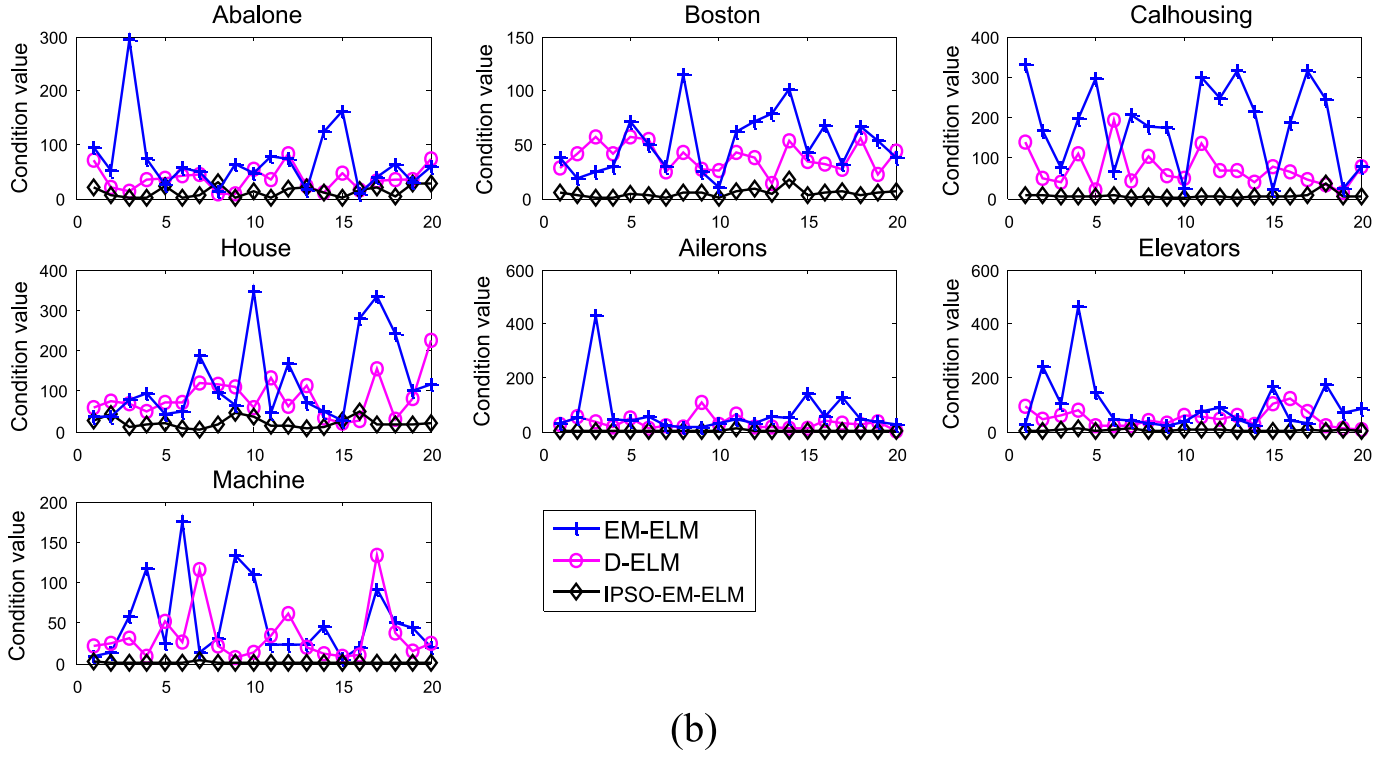


Fig. 2. (a) The condition values of the hidden output matrix in the five ELM with 20 trials (b) The condition values of the hidden output matrix in the EM-ELM,D-ELM and IPSO-EM-ELM with 20 trials.

$$p_g = \begin{cases} p_i & (f(p_g) - f(p_i) > \eta f(p_g)) \ or \\ & and\ (K_i < K_g)) \\ p_g & else \end{cases} \qquad (f(p_g) - f(p_i) < \eta f(p_g) \tag{17}$$

where $f(p_i)$,$f(p_{ib})$and $f(p_g)$ are the corresponding fitness values for the ith particle, the best position of the ith particle and global best position of all particle, respectively; η >0 is the tolerance rate; K_i,K_{ib} and K_g are the condition values of the hidden output matrix of the SLFNs related to the ith particle, the best position of the ith particle and global best position of all particle, respectively. According to the literature [19,20], the 2-norm condition value of the matrix **H** can be computed as Eq. (18):

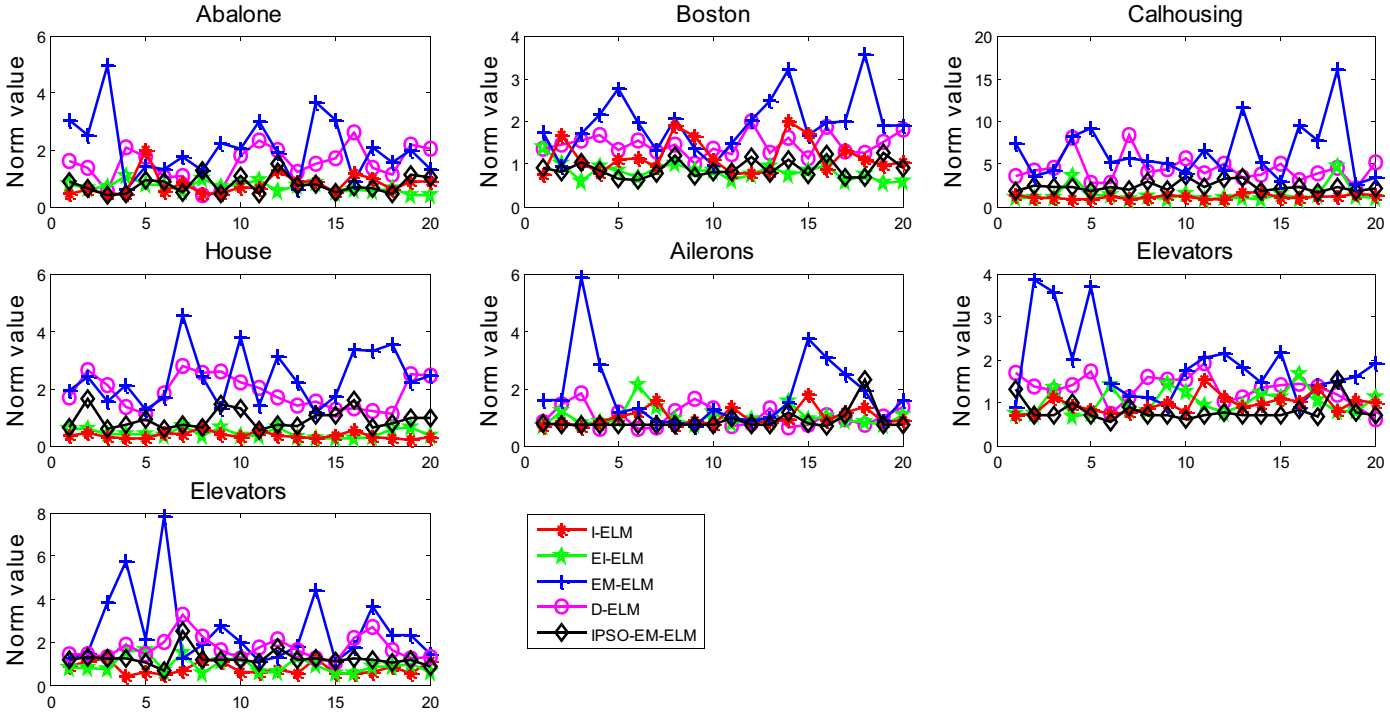


Fig. 3. The 2-norm values of output weights in the five ELM with 20 trials.

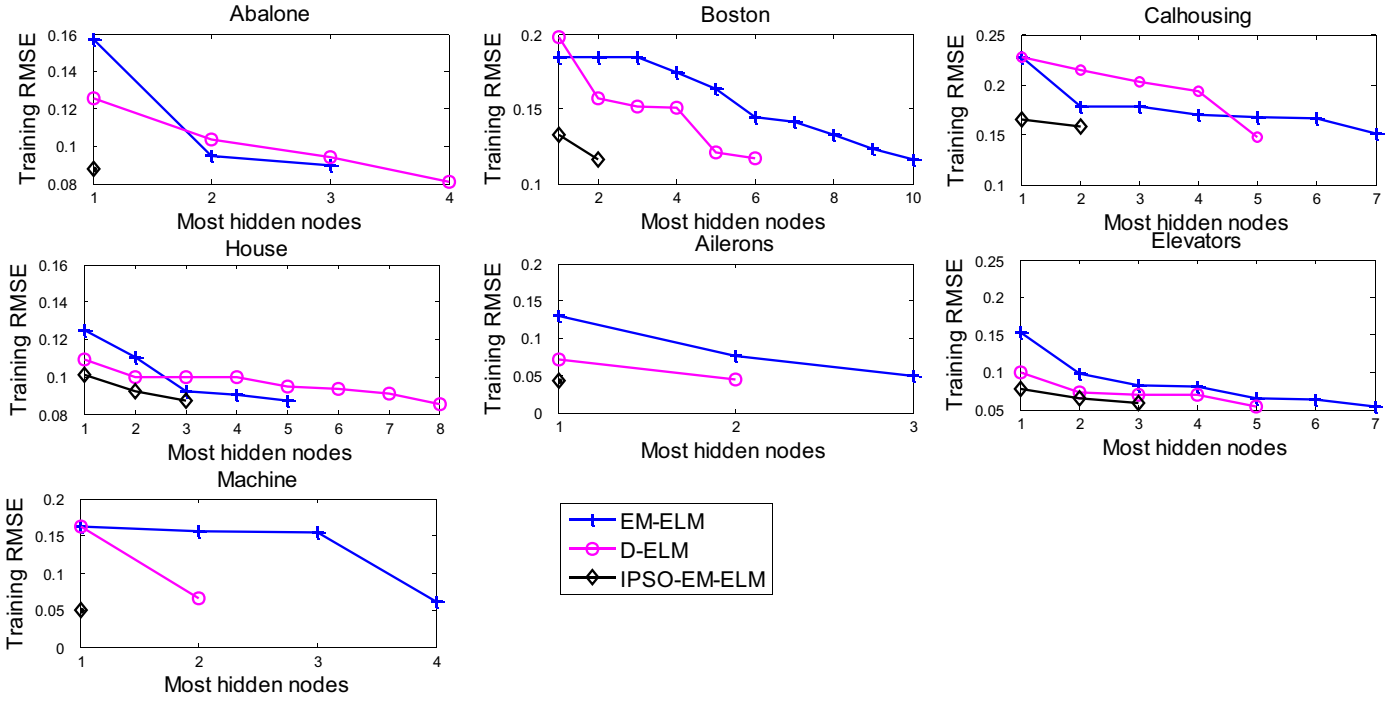


Fig. 4. The convergence curves of three constructive ELM.

$$K_2(\mathbf{H}) = \sqrt{\frac{\lambda_{\max}(\mathbf{H}^T\mathbf{H})}{\lambda_{\min}(\mathbf{H}^T\mathbf{H})}} \tag{18}$$

where $\lambda_{\max}(\mathbf{H}^T\mathbf{H})$ and $\lambda_{\min}(\mathbf{H}^T\mathbf{H})$ are the largest and the smaller eigenvalues of the $\mathbf{H}^T\mathbf{H}$.

– Substep 2.4: Each particle updates its position according to Eqs. (12) and (13). According to [20,24], all components in the particle should be limited within the range of [−1, 1].
– Substep 2.5: Repeat the above substeps until the goal is met or the maximum epoch is completed. Then the optimal weights and hidden bias($\mathbf{a}_{L*}$, b_{L*}) are obtained from the global best of the swarm.

– Step 3: According to Eq.(4) to calculate the output weight of the SLFN after add the new hidden node($\mathbf{a}_{L*}$, b_{L*})as $L=1$. According to Eq. (9) to calculate the output weight of the network after add the new hidden node($\mathbf{a}_{L*}$, b_{L*})as $L>1$;

– Step 4: Calculate the output error $E(\mathbf{H}_L) = \|\mathbf{H}_L\mathbf{H}_L^{+}\mathbf{T} - \mathbf{T}\|$;
– Step 5: Go to Step 1 until $L > L_{\max}$ or $E(\mathbf{H}_L) < \varepsilon$, and the

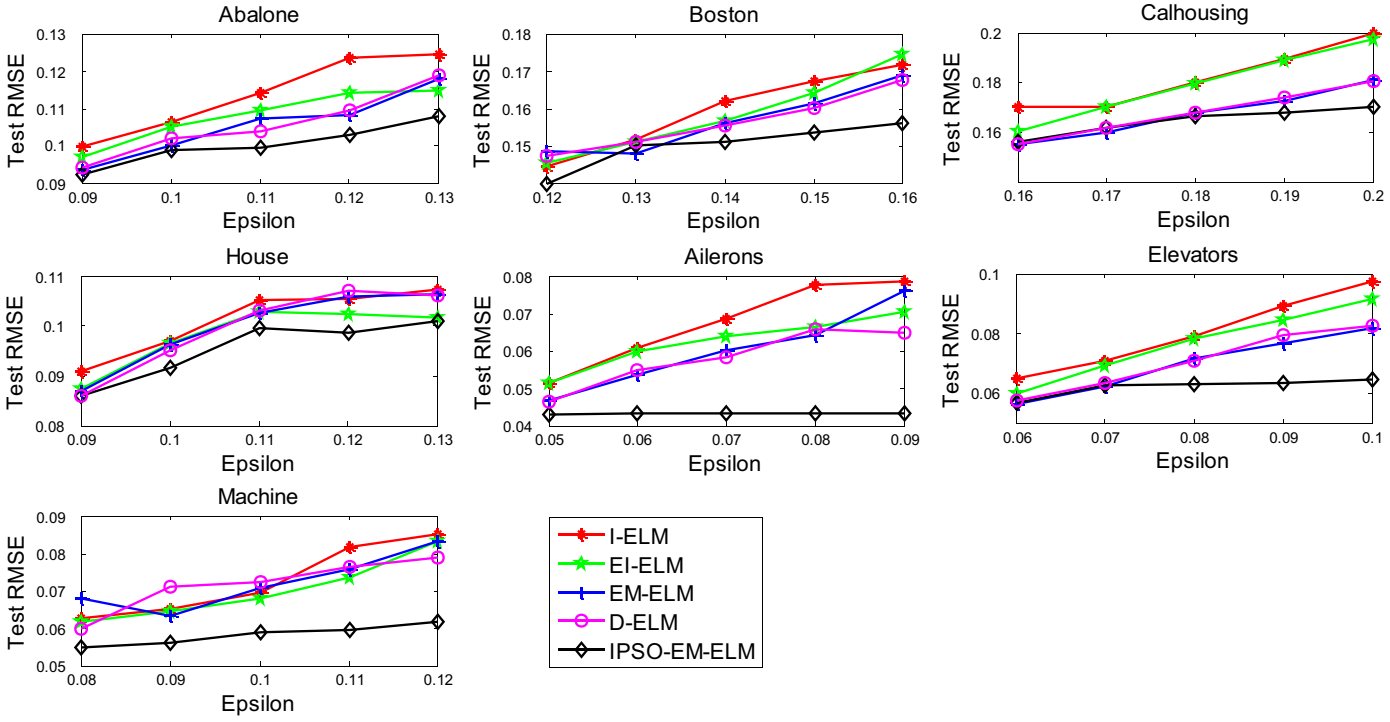


Fig. 5. Testing RMSE versus the epsilon on the seven data sets.

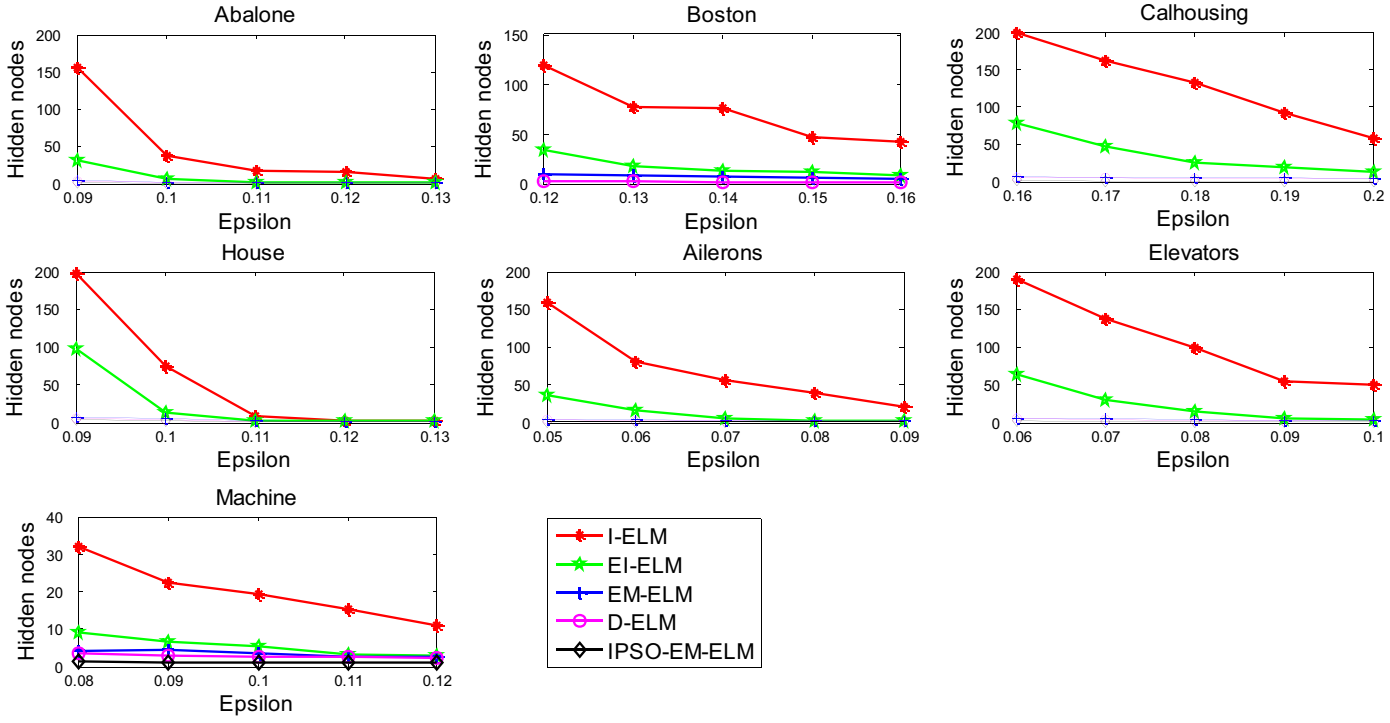


Fig. 6. The number of the hidden nodes versus the epsilon on the seven data sets.

constructed ELM is obtained.

4. Experiment results and discussion

To verify the effectiveness and efficiency of the proposed algorithm, IPSO-EM-ELM is compared with other classical constructive ELM including I-ELM [12], EI-ELM [14], EM-ELM [15] and D-ELM [17] on seven benchmark regression problems from UCI database [28]. The specifications of the problems are shown in Table 1.

In the experiments, the inputs and outputs have been normalized in the range of [−1, 1] and [0,1], respectively. For each trial of simulations, the data set of the application was divided into training and testing data sets according to Table 1. The sigmoid function $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a}\cdot\mathbf{x} + b)))$ is selected as the activation function of the hidden layer in all algorithms. The hidden node parameters $\boldsymbol{a}$ and b are randomly chosen from the range of [−1,1] based on a uniform sampling distribution probability. In the experiments, some parameters including the population size, maximum iteration number,

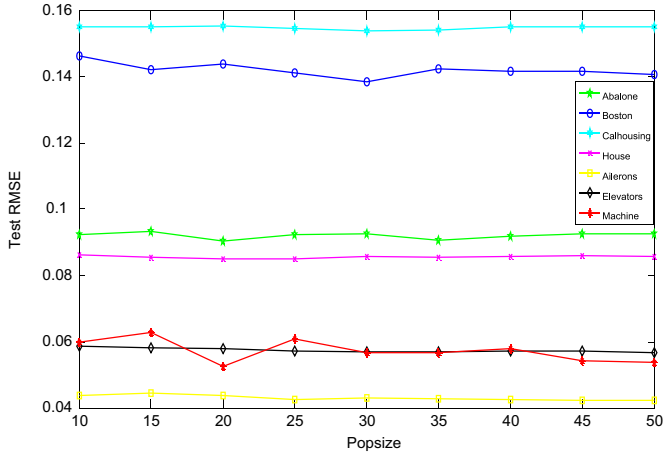


Fig. 7. Testing RMSE versus the popsize on the seven data sets.

and tolerance rate, in IPSO-EM-ELM are determined by trial and error. The population size and maximum iteration number are set as 30 and 10, respectively; the tolerance rate is selected as 0.02. According to [27], the initial inertial weight w_{ini} and the final inertial weight w_{end}, are selected as 0.9 and 0.4 respectively; the acceleration constants c_1 and c_2 are both selected as 1.6. All the results shown in this paper are the mean values of 20 trials. All the experiments are run in MATLAB R2012a environment and the same PC with Intel Core 2 Duo 2.93 GHZ CPU and 2GB RAM.

Table 2 shows the average test RMSE, and the training time of I-ELM, EI-ELM, EM-ELM, D-ELM, IPSO-EM-ELM. The IPSO-EM-ELM obtains the least test RMSE of all ELM, which indicates the proposed method has the better generalization performance than the other four ELM. However, the proposed method requires most time to train SLFN than other ELM, since the IPSO-EM-ELM use PSO to select the optimal hidden nodes.

Table 3 shows the final number of hidden nodes, condition value of the hidden output matrix ($K(H)$) and norm value of the output weights in different ELM. From Table 3, the number of the hidden nodes and the condition value in IPSO-EM-ELM are both the least in all cases among the five ELM, and the norm value of IPSO-EM-ELM is less than EM-ELM and D-ELM for all data sets. Therefore, the IPSO-EM-ELM could establish more compact and well-conditioned SLFN than other ELM.

Fig. 2 shows the corresponding condition value of the hidden output matrix in the five ELM on seven data sets. From Fig. 2(a), the condition values in the EM-ELM, D-ELM, IPSO-EM-ELM are much smaller than I-ELM and EI-ELM. From Fig. 2(b), the condition value in IPSO-EM-ELM is smaller than that in EM-ELM and D-ELM.

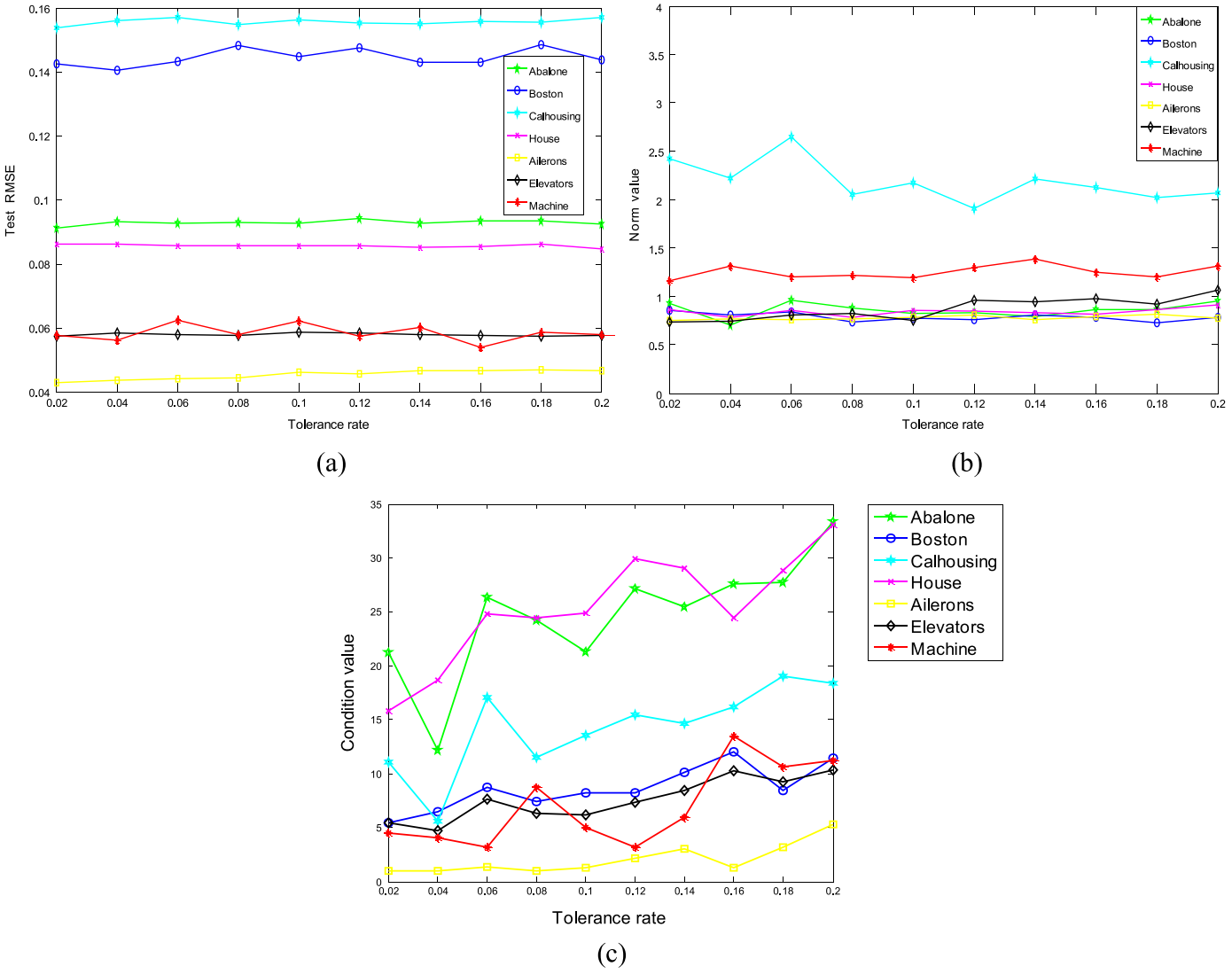


Fig. 8. The effects of tolerance rate in the IPSO-EM-ELM on the seven data sets with (a) Test RMSE, (b) Norm value, (c) Condition value.

Moreover, the condition curve for IPSO-EM-ELM is more stable than the other four ELM. Therefore, the SLFN established by the IPSO-EM-ELM is better-conditioned than the ones constructed by the other four ELM.

Fig. 3 shows the corresponding 2-norm values of five ELM on seven data sets. From Fig. 3, the norm values of the output weights obtained by I-ELM, EI-ELM, IPSO-EM-ELM are less than that of the EM-ELM and D-ELM. The I-ELM achieves the smallest norm value in Calhousing and House cases, while the norm values of I-ELM, EI-ELM, and IPSO-EM-ELM are very close in other five data sets. Therefore, the IPSO-EM-ELM could achieve better generalization than the EM-ELM and D-ELM.

Fig. 4 shows the convergence curves of three constructive ELM including EM-ELM, D-ELM, and IPSO-EM-ELM. Obviously, the IPSO-EM-ELM requires much less hidden nodes with less training RMSE than the EM-ELM and D-ELM.

Fig. 5 and Fig. 6 show the relationship between the epsilon (stop training RMSE) with the test RMSE and the number of the hidden node in five ELM in all cases, respectively. With the increase of the epsilon, the test RMSE has an upward trend and the required number of hidden nodes has a downward trend in all cases for all ELM. With the same epsilon, the IPSO-EM-ELM could obtain less test RMSE with fewer hidden nodes than other ELM nearly in all cases.

Fig. 7 shows the relationship between the popsize and the test RMSE for the IPSO-EM-ELM. The test RMSE has a slight downward trend on the Boston, Ailerons, Elevators and Machine data when the popsize increases, while the test RMSE is not sensitive to the popsize on the other three data.

Fig. 8 shows the effects of the tolerance rate, η, on the test RMSE, condition value of the hidden output matrix and 2-norm value of the output weights in the IPSO-EM-ELM on seven datasets. The tolerance rate is selected from [0.02,0.2] at identically spaced intervals. From Fig. 8(a), in Boston, Calhousing, and Machine cases, the test RMSE fluctuate with the increase of the tolerance rate, while it has not changed significantly as the tolerance rate increase on the other datasets. From Fig. 8(b), the norm value is not much sensitive to the tolerance rate in most calses. From Fig. 8(c), with the increase of the tolerance rate, the condition value has an upward trend in all cases.

5. Conclusions

In this paper, an incremental error minimization extreme learning machine based on particle swarm optimization (IPSO-EM-ELM) was proposed. Different from traditional incremental error minimization extreme learning machine, the proposed method selected the hidden nodes with PSO by considering not only the training RMSE but also the condition performance of the SLFN, so the SLFN established by the proposed method was compact and well-conditioned with better generalization performance. Because of using PSO to select the weight parameters, IPSO-EM-ELM inevitably spent more time than other ELM. Future research work will include how to solve this problem and apply the IPSO-EM-ELM to complex classification problems.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Nos. 61271385, 61572241 and 61472138), the Foundation of the Peak of Six Talents of Jiangsu Province (No. 2015-DZXX-024), and the Fifth "333 High Level Talented Person Cultivating Project" of Jiangsu Province.

References

[1] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine,, IEEE Trans. Neural Netw. 21 (1) (2010) 158–162.

[2] D.S. Huang, A constructive approach for finding arbitrary roots of polynomials by neural networks,, IEEE Trans. Neural Netw. 15 (2) (2004) 477–491.

[3] D.S. Huang, Horace H.S.Ip, Zheru Chi, " A neural root finder of polynomials based on root moments, Neural Comput. 16 (8) (2004) 1721–1762.

[4] D.S. Huang, Horace H.S.Ip, Law Ken C K, Zheru Chi, "Zeroing polynomials using modified constrained neural network approach,", IEEE Trans. Neural Netw. 16 (3) (2005) 721–732.

[5] Y.H. Pao, S.M. Phillips, D.J. Sobajic, Neural-net computing and the intelligent control of systems,, Int. J. Control 56 (2) (1992) 263–289.

[6] G.-B.Huang, Q.-Y.Zhu, C.-K.Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings International Joint Conference Neural Netwvol. 2, pp. 985–990, 2004.

[7] Yong-Ping Zhao, Ramón Huerta, Improvements on parsimonious extreme learning machine using recursive orthogonal least squares,, Neurocomputing 191 (2016) 82–94.

[8] Min Han, Xue Yang, Enda Jiang, An extreme learning machine based on Celluar Automata of edge detection for remote sensing images,, Neurocomputing 198 (2016) 27–34.

[9] G.-B.Huang, C.-K.Siew, Extreme learning machine: RBF network case, in: Proceedings of the 8th International Conference Control Autom. Robot. Visvol. 2, pp. 1029–1036, 2004.

[10] G.-B.Huang, N.-Y.Liang, H.-J.Rong, P.Saratchandran, N.Sundararajan, On-line sequential extreme learning machine, in: Proceedings IASTED International Conference Comput. Intell., Calgary, AB, Canada, 2005.

[11] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks,, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423.

[12] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes,, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.

[13] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications,, Neurocomputing 70 (1–3) (2006) 489–501.

[14] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine,, Neurocomputing 71 (16–18) (2008) 3460–3468.

[15] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning,, IEEE Trans. Neural Netw. 20 (8) (2009) 1352–1357.

[16] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, Universal approximation of extreme learning machine with adaptive growth of hidden nodes,, IEEE Trans. Neural Netw. Learn. Syst. 23 (2) (2012) 365–371.

[17] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, Yeng Chai Soh, Dynamic extreme learning machine and its approximation capability,, IEEE Trans. Cybern. 43 (2013) 2054–2065.

[18] G.Zhao, Z.Shen, C.Miao, R..Gay, Enhanced extreme learning machine with stacked generalization, in: Proceedings International Joint Conference on Neural Networkspp. 1192–1199, 2008.

[19] G.P. Zhao, Z.Q. Shen, C.Y. Miao, Z.H. Man, On improving the conditioning of extreme learning machine: a linear case, in: Proceedings of the 7th International Conference on Information, Communi-cations and Signal Processing, 2009 (ICICS2009), Maucu, China, December 8–10, 2009, pp. 1–5.

[20] F. Han, H.-F. Yao, Q.-H. Ling, An improved evolutionary extreme learning machine based on particle swarm optimization,, Neurocomputing 116 (2013) 87–93.

[21] J.Kennedy, R.Eberhart, Particle , Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, vol. 4, 1995, pp. 1942–1948.

[22] Daniel Parrott, Xiao dong Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput. 10 (4) (2006) 440–458.

[23] W.B. Langdon, Riccardo Poli, Evolving problems to learn about particle swarm optimizers and other search algorithms,, IEEE Trans. Evol. Comput. 11 (5) (2007) 561–578.

[24] You Xu, Yang Shu, Evolutionary extreme learning machine-based on particle swarm op-timization, in: International Symposium on Neural Networks 2006 (ISNN2006), LNCS, vol. 3971, 2006, pp. 644–652.

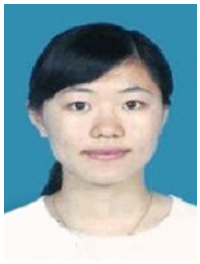
[25] Fei Han, Min-Ru Zhao, Jian-Ming Zhang, An improved incremental error minimized extreme learning machine for regression problem based on particle swarm optimization,, Lect. Notes Comput. Sci. 9927 (2015) 94–100.

[26] R.C. Eberhart, J. Kennedy, A new optimizer using particles swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human science, Nagoya, Japan, 1995, pp. 39–43.

[27] Y.Shi, R.C.Eberhart, A modified particle swarm optimizer, in: Proceedings of IEEE World Conference on Computation Intelligence, 1998, pp. 69–73.

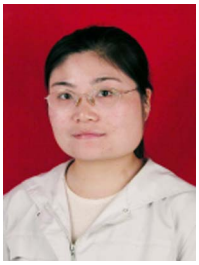
[28] C.L.Black, C.J.Merz, UCI Repository of Machine Learning Databases. Dept. Inf. Comput. Sci., Univ. California, Irvine, ⟨http://www.ics.uci.edu/~mlearn/mlrepository.html⟩.

Fei Han received the M.A. degree from Hefei University of Technology in 2003 and the Ph.D. degree from University of Science and Technology of China in 2006. He is currently a professor of computer science at Jiangsu University. His principal research interests are intelligent computing and intelligent information processing, including neural networks, particle swarm optimization, and bioinformatics.



Min-Ru Zhao received the B.S. and the M.A. degree from Jiangsu University in 2011 and 2015. She is currently an engineer of Kunshan branch of Industrial and Commercial Bank of China. Her research interests include neural networks and intelligent information processing.

Jian-Ming Zhang received the B.S. degree from Nanjing University in 1986, the M.A. degree from Jiangsu University of Science and Technology in 1997 and the Ph.D. degree from Nanjing University of Science and Technology in 2005. He is currently a professor of computer science at Jiangsu University. His principal research interests are pattern recognition and image processing.



Qing-Hua Ling received the B.S. degree from Nanjing Normal University in 2002 and the M.A. degree from Hefei Institute of Intelligent Machines, Chinese Academy of Sciences in 2005. She is currently a lecturer of computer science at Jiangsu University of Science and Technology. Her research interests include neural networks, particle swarm optimization and bioinformatics.