www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 8 Issue 3 March 2019, Page No. 24491-24501
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v8i03.4294

Improve Efficiency of Extreme Learning Machine based on Chaos Particle Swarm Optimization Method

Yuwen Pan, Zhan Wen*, Yahui Chen, Wenzao Li*

School of Communication Engineering, Chengdu University of Information Technology, Chengdu, Sichuan; 610225, China;
Corresponding Author: **Yahui Chen, Wenzao Li**

Abstract
Extreme Learning Machine (ELM) and Regularized Extreme Learning Machine (RELM) have advantages of fast training speed and good generalization. However, ELM/RELM often needs numerous number of hidden layer nodes to get better performance. The superabundant nodes in hidden layer maybe lead to low running speed. Thus it is not feasible to use ELM in some fields that require high speed algorithms. Therefore, in this paper, we propose an Improved ELM/RELM Optimized based on Chaos Particle Swarm Optimization (CPSO-ELM/RELM) to reduce the number of hidden layer nodes, but still maintain a desirable accuracy. At the same time, it lowers the running speed compared with other algorithms. To verify the application of this method, we design numerous experiments for ELM and RRELM. Their simulation shows that the approach improves the speed of the algorithms, and the accuracy is still high. This makes it possible to use improved CPSO-ELM/RELM in some system with high real-time requirements.
Keywords: extreme learning machine; Regularized Extreme Learning Machine; particle swarm optimization algorithm; running speed

1. Introduction

1.1 Extreme Learning Machine (ELM)

Traditional feedforward neural networks such as BP neural network has the disadvantages of slow training speed, bad generalization and easy to fall into local minimum solution, etc. It may take a lot of time to train but get unsatisfactory results. So, Huang et al proposed a new type of feedforward neural network called extreme learning machine to solve such problems[1]. Unlike BP[2], it randomly generates the input matrix and bias matrix of hidden layer, and there is no adjustment during the training process. Only need to set the number of nodes in the hidden layer. Compared with BP, its training speed and generalization are greatly improved so it can be widely used in many fields such as medicine, face recognition and regression analysis.

1.2 Regularized Extreme Learning Machine(RELM)

Since the method of regularization was proposed[3], it has become one of the core concepts of neural networks and machine learning algorithms. DENG et al. applied the regularization method to the ELM[4]. RELM keeps the advantage of fast training speed and can achieve the better performance.

1.3 PSO-ELM/RELM

Due to the random parameters (number of hidden layer nodes, input matrix and bias matrix of hidden layer), the stability of ELM/RELM is poor. To improve it, some researchers propose that using Particle Swarm

Optimization (PSO) algorithm to optimize these random parameters can promote the stability and accuracy of the results[5][6].

However, to some which already have the good performance, PSO-ELM/RELM can't obviously improve its stability and accuracy. So, we try to improve its running speed. But ELM/RELM usually need a large of number of hidden layer nodes to get a good performance, which will greatly increase the complexity of the network structure and take up much time to get the results. Based on this problem, we propose a method to solve it. First, to reduce the number of hidden layer nodes and promote the speed of ELM/RELM, we use the empirical formula $L = \sqrt{n+m} + \omega$ to select L(the number of hidden layer nodes). Where n is the number of input layer nodes, m is the number of output layer nodes, and $1 \leq \omega \leq 10$ is a random number[7]. (For ELM and RELM, the number of hidden layer nodes selected by the empirical formula is usually less than the best solution which is obtained by CPSO algorithm, whose purpose is to reduce the number. And its accuracy is slow. These will be proved in section 4). Second, because of the low precision caused by the reduction of number of hidden layer nodes, we use CPSO algorithm to optimize the input matrix and bias matrix of hidden layer to improve the accuracy of the results. Through the above steps, we can get the model, improved CPSO-ELM/RELM, which has the less hidden layer nodes, faster speed and high enough accuracy and make it possible to solve those problems that require both precision and timeliness.

There is two main contributions in this paper. We propose Improved CPSO-ELM/RELM algorithms. First, the number of hidden layer nodes can be reduced through $L = \sqrt{n+m} + \omega$ instead of PSO algorithm, but still maintain a relatively high accuracy. Second, the algorithms obviously promote running speed based on a relatively high accuracy.

The rest of the paper is organized as follows: In Section 2, we will introduce ELM, RELM, PSO and CPSO. Section 3 presents the system model and algorithm steps. Finally, we give simulation results in the Section 4.

2. Related Works

Generally speaking, ELM is used for classification and regression in many fields[8]. And RELM continues to improve ELM to make it have a better generalization and accuracy. PSO and CPSO can further improve the performance of ELM/RELM. In this section, we will introduce the theory and algorithm formulas of ELM, RELM and algorithm steps first. Then PSO and CPSO will be introduced.

2.1 ELM and its Specific Algorithm Formulas

Extreme Learning Machine (ELM) is a kind of machine learning algorithm based on feedforward neuron network. Its main feature is that the hidden layer parameters can be random or artificially given and do not need to be adjusted. The learning process only needs to calculate the output weight. ELM has the advantages of high learning efficiency and strong generalization ability, and is widely used in classification, regression, clustering, feature learning and other fields. Compared with the traditional algorithms such as BP neural networks, ELM algorithm has advantages of faster learning speed and better generalization performance. The model of ELM is shown as follow:

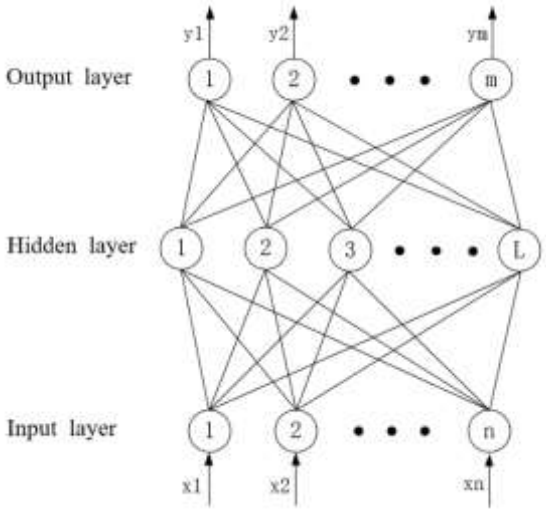

Figure 1 feedforward neural network model

Supposing there is a training set with *N* samples, the input matrix and output matrix are as follow:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nN} \end{bmatrix}_{n\times N} \quad (1)$$

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m1} & \cdots & y_{mN} \end{bmatrix}_{m\times N} \quad (2)$$

Where X and Y represent input matrix and output matrix respectively.

Assuming that connection weight matrix between the input layer and the hidden layer is *W*:

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1} & w_{L2} & \cdots & w_{Ln} \end{bmatrix}_{L\times n} \quad (3)$$

Where w_{ij} is the connection weight between the *i-th* node of the hidden layer and the *j-th* node of the input layer.

The connection weight matrix between the hidden layer and the output layer is β:

$$\beta = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{L1} & \beta_{L2} & \cdots & \beta_{Lm} \end{bmatrix}_{L\times m} \quad (4)$$

Where β_{ij} represents the connection weight between the *i-th* node of the hidden layer and the *j-th* node of the output layer.

Supposing the bias weight matrix is b:

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_L \end{bmatrix}_{L\times 1} \quad (5)$$

Suppose that the activation function of the hidden layer is $g(x)$. Through the calculation formula of feedforward neuron network and Figure 1, we can obtain the hidden layer output matrix is:

$$H = \begin{bmatrix} g(w_1 x_1 + b_1) & g(w_2 x_1 + b_2) & \dots & g(w_L x_1 + b_L) \\ g(w_1 x_2 + b_1) & g(w_2 x_2 + b_2) & \dots & g(w_L x_2 + b_L) \\ \vdots & \vdots & \ddots & \vdots \\ g(w_1 x_N + b_1) & g(w_2 x_N + b_2) & \dots & g(w_L x_N + b_L) \end{bmatrix}_{N \times L} \quad (6)$$

In order to minimize the error between the actual calculated value $H\beta$ and the target value Y^T, we can get:

$$\beta = min_\beta \|H\beta - Y^T\| \quad (7)$$

So, the connection matrix between the hidden layer and the output layer is:

$$\beta = H^+ Y^T \quad (8)$$

Where H^+ represents the *Moore-Penrose* generalized inverse matrix of the hidden layer output matrix *H*.

2.2 RELM and its Specific Algorithm Formulas

Since the method of regularization was proposed, it has become the one of the most popular method in the field of machine learning [9]. ELM is based on Empirical Risk Minimization (ERM) principle and so it tends to overfit. According to statistical learning theory, to make ELM have better generalization, we should consider both Structural Risk Minimization (SRM) and ERM. The model should have the best trade-off between the two risks. RELM is mainly by weighting the most small squares evaluation loss function to obtain output weight:

$$\min_\beta f(\beta) = \|H\beta - Y\|^2 + C\|\beta\|^2 \quad (9)$$

Where $C > 0$ is the compromise between $\|H\beta - Y\|^2$ and $\|\beta\|^2$. $\|H\beta - Y\|^2$ and $\|\beta\|^2$ are ERM and SRM respectively. Differentiate for β:

$$\frac{\partial f(\beta)}{\partial \beta} = -2H^T(Y - H\beta) + 2C\beta \quad (10)$$

Let the derivative equal 0:

$$2H^T H\beta - 2H^T Y + 2C\beta = 0 \quad (11)$$

We can get:

$$(CI + H^T H)\beta = H^T Y \quad (12)$$

So the connection weight matrix between hidden layer and output layer can be expressed as:

$$\beta = \begin{cases} (CI + H^T H)^{-1} H^T Y, & N \geq L \\ H^T (CI + H^T H)^{-1} Y, & N < L \end{cases} \quad (13)$$

2.3 ELM/RELM Algorithm Steps

The above introduces the basic theory of ELM and RELM. The following content will present the algorithm steps.

1. Randomly generating connection weight matrix between the input layer and the hidden layer W and bias matrix b.
2. Calculating hidden layer output matrix *H* by Eq(6).
3. Obtaining connection weight matrix between the hidden layer and the output layer β by Eq(8) or Eq(13).
4. Through above steps, we can get the model of ELM/RELM.

PSO algorithm is a global optimization algorithm. It has the edge of fast convergence speed, which makes it feasible to be used to solve the high dimensional problem. However, PSO algorithm is easy to fall into local optimal solution. Therefore, chaos algorithm is introduced to help PSO algorithm search global optimal solution, which called CPSO algorithm. PSO and CPSO algorithm will be introduced as follow.

2.4 PSO and its Specific Algorithm Formulas

PSO is an evolutionary computation that was proposed in 1995 by Dr. Eberhart and Dr. Kennedy from a behavioral study of predation of birds. The algorithm was originally inspired by the regularity of the bird cluster activity, and then a simplified model built using group intelligence. Based on the observation of the activity behavior of animal clusters, the particle swarm optimization algorithm uses the individual's sharing of information in the group to make the movement of the whole group in the problem solving space from the disordered to the orderly evolution process, so as to obtain the optimal solution. All particle has only two properties: velocity *v* and position *x*, velocity is the speed of every particle, and position represents the place of every particle. Each particle searches for the optimal solution separately in the search space and records it as the current individual optimal value P_{best}, and shares the individual optimal value with other particles to find the current global optimal solution G_{best}. All particles in the particle swarm adjust their velocity and position according to P_{best} and G_{best}. After velocity and position of each particle are updated, the velocity and position of this particle need to be limited to avoid exceeding the search range[10][11].

The speed update formula for the *d-th* dimension of particle *i*:

$$V_{id}^{k+1} = \omega V_{id}^{k} + c_1 r_1 \left(P_{id}^{k} - X_{id}^{k}\right) + c_2 r_2 (P_{gd}^{k} - X_{id}^{k}) \quad (14)$$

Where c_1 and c_2 are the acceleration constants. r_1 and r_2 are the random numbers from 0 to 1. ω is the inertia factor. P_{id}^{k} is the current individual optimal value, and P_{gd}^{k} is current global optimal solution.

The position update formula for the *d-th* dimension of particle *i*:

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k} \quad (15)$$

2.5 CPSO and its Specific Algorithm Formulas

2.5.1 Chaos Algorithm

Chaos is a characteristic of a nonlinear system. It demonstrates the dependence on initial conditions and the infinitely unstable periodic motion. Due to its non-repetitive nature, it can perform a comprehensive search. Chaos algorithm is usually used in conjunction with other optimization algorithms to obtain the global solution. The following gives a rule of chaotic motion, the formulas to map from particle to chaotic variable is:

$$cx_i^1 = (x_i - pmin)(pmax - pmin) \quad (16)$$

Where cx_i^1 is the chaotic variable of *i-th* particle, x_i is *i-th* particle, $pmin$ is the minimum value of searching area, $pmax$ is the maximum value of the searching area.

$$cx_i^{(k+1)} = 4cx_i^k(1 - cx_i^k) \quad (17)$$

Where cx_i^k is the value of cx_i^1 after chaotic motion in step *k*.

The way to map from chaotic variable to x_i is:

$$x_i = pmin + cx_i(pmax - pmin) \quad (18)$$

2.5.2 CPSO Algorithm Theory and Steps

Traditional PSO algorithm has the disadvantage of being easily trapped in local optimal solution. When a particle find a good solution, other particles will be attracted to approach it, which makes it difficult to jump out of the current optimal solution. In another word, it is easy to fall into local optimal solution. To solve this problem, CPSO is proposed. When there are too many particles accumulating together, chaos algorithm will disrupt these particle by Eq(16), Eq(17) and Eq(18). Then comparing the particles after disruption with particles before disruption(new particles and original particles), if the fitness of new is better than the original, replacing the original with the new, otherwise, reserve the original. It makes the position of

particles not only be adjusted by Eq(14) and Eq(15) but also by chaotic motion, which can ensure population diversity and the convergence rate both and makes every particle try to traverse the entire searching area to obtain the global optimal solution[12]. The specific algorithm steps are as follow:

1. Generating the particles as stated in **2.4.**
2. Changing their velocity *v* and position *x* by Eq(14) and Eq(15).
3. Judging if there is no obvious change to G_{best} in the last 10 iterations. If not, perform 4, otherwise, do as follow:

   ① Mapping the position of original space to chaotic space by Eq(16)

   ② Adjusting the position of every particle by Eq(17).

   ③ Mapping the position of chaotic space to original space by Eq(18).

   ④ Updating the new population.

4. Determining if the output condition is met, if not, return to 2, otherwise, output the results.

3. Algorithm Steps

As mentioned above, the ELM/RELM needs too many hidden layer nodes to get the good performance and so, the speed is slow. Therefore, we use the empirical formula $L = \sqrt{n+m} + \omega$ to select the number of hidden layer nodes, which can reduce the number of hidden layer nodes and boost speed. However, when the number of hidden layer nodes is too little, the performance of ELM/RELM will be bad. Thus, we use CPSO algorithm to optimize the random parameters, input matrix and bias matrix. It can greatly improve the stability and accuracy of the results. Through above steps, we can get the model with faster speed and the high precision enough.

3.1 Experiments Method

To prove that this method is suitable for both ELM and RELM, we use ELM and RELM to do the experiments separately. First, we divide data sets into three parts. One is the training set and the others are two testing sets. The first testing set T_{test1} is used to adjust the parameters with CPSO algorithm, and the second testing set T_{test2} is used to calculate the accuracy of results. Second, we use empirical formula $L = \sqrt{n+m} + \omega$ and CPSO algorithm to select the number of hidden layer nodes separately. Finally, input matrix and bias matrix will be optimized by CPSO algorithm. We call the model whose number of hidden layer nodes is chosen by CPSO algorithm CPSO-ELM/RELM, and another is called improved CPSO-ELM/RELM.

3.2 Pseudo Code

3.2.1 Pseudo Code of Improved CPSO-ELM/RELM

It's worth noticing that because the data set is too small, we perform 100 experiments and randomly divide the training set and testing set before every experiment. Because when the number of hidden layer nodes is optimized, input matrix and bias matrix of hidden layer are random, training set and testing set are also randomly divided, so the result is random. Based on this problem, for each particle, we will do 50 experiments and use their sum as the fitness. However, when the input matrix and bias matrix of hidden layer are optimized, although the number of hidden layer nodes is determined, the division of training set and testing set are still random. And so, for each particle, we also need to do more than one experiment. For the speed of calculation, we set it to 3.

Improved CPSO-ELM/RELM

1. Dividing the training set and double testing set.
2. Using empirical formula $L = \sqrt{n+m} + \omega$ to select the number of hidden layer nodes.
3. Using training set and first testing set to optimize the input matrix and output matrix of hidden layer with CPSO algorithm(Input matrix and output matrix of hidden layer is seen as a particle):

   ① Initialize important parameters(c_1, c_2, ω, *M*) and normalize the input of training set and testing set.

   ② Calculating fitness of all particles, the steps are as follow:

   for *j* = 1:*M*
   ρ_j =0;
   for *i* = 1:3
   Randomly disrupting the training set and first testing set.
   Calculating accuracy followed by **2.3**, record as ε;
   $\rho_j = \rho_j + \varepsilon$;
   end

   ③ ρ is the fitness of particles. Calculating P_{best} and G_{best}, updating the velocity and position of the particles according to (14) and (15). And limit the velocity and position of the particles.

   ④ Calculating the fitness of new particles and update P_{best} and G_{best}.

   ⑤ Operating the particles as described in **2.5.2**.

   ⑥ If G_{best} does not change after 10 iterations, it will jump out of the loop, otherwise, return to ②.

4. Through the above step, we can get the model of improved CPSO-ELM/RELM.

3.2.2 Pseudo Code of CPSO-ELM/RELM

CPSO-ELM/RELM

1. Dividing the training set and double testing set.

2. Using training set and first testing set to optimize the number of hidden layer nodes with CPSO algorithm(The number of hidden layer nodes is seen as a particle):

   ① Initialize important parameters(c_1, c_2, ω, *M*) and normalize the input of training set and testing set.

   ② Calculating fitness of all particles, the steps are as follow:

   for *j* = 1:*M*
   ρ_j =0;
   for *i* = 1:50
   Randomly disrupting the training set and first testing set.
   Calculating accuracy, record as ε;
   $\rho_j = \rho_j + \varepsilon$;
   end

   ③ ρ is the fitness of particles. Calculating P_{best} and G_{best}, updating the velocity and position of the

particles according to (14) and (15). And limit the velocity and position of the particles.

④ Calculating the fitness of new particles and update P_{best} and G_{best}.

⑤ Operating the particles as described in **2.5.2**.

⑥ If G_{best} does not change after 10 iterations, it will jump out of the loop, otherwise, return to ②.

3. Refer **3.2.1** to optimize input matrix and bias matrix of hidden layer.

4. Through the above step, we can get the model of CPSO-ELM/RELM.

4. Simulation And Results

In this section, we will introduce the simulation environment, important parameter settings and final results. Final results will be divided into two parts. First part is going to give the accuracy of ELM/RELM, CPSO-ELM/RELM and improved CPSO-ELM/RELM. And second part will give the results of the running time of ELM/RELM, CPSO-ELM/RELM and improved CPSO-ELM/RELM. We will compare the results and display them in the form of pictures and tables.

4.1 Simulation Environment and Important Parameter Settings.

The simulation environment is in MATLAB, and the important parameter settings are as follow (They are set based on empirical values):

Table 1 The settings of important parameter

parameter	value
c_1	2.05
c_2	2.05
ω	1

4.2 The Results of Accuracy of ELM/RELM, CPSO-ELM/RELM and Improved CPSO-ELM/RELM.

From data information above, we can see that the number of input layer nodes is 9 and the number of output layer nodes is 1. So we can get that for ELM/RELM and improved CPSO-ELM/RELM. We select the number of hidden layer as 13 according to $L = \sqrt{n+m} + \omega$. For CPSO-ELM/RELM, we use **3.2.2** to obtain the number. From the simulation results, the number of hidden layer nodes of CPSO-ELM/RELM is 77, 324. After getting the number, we do the experiments following section 3. In order to eliminate the effects of too small data setting and acquire the more accurate results, we do 100 experiments and disrupting the training set and testing set for every experiment. The results are shown as follow:

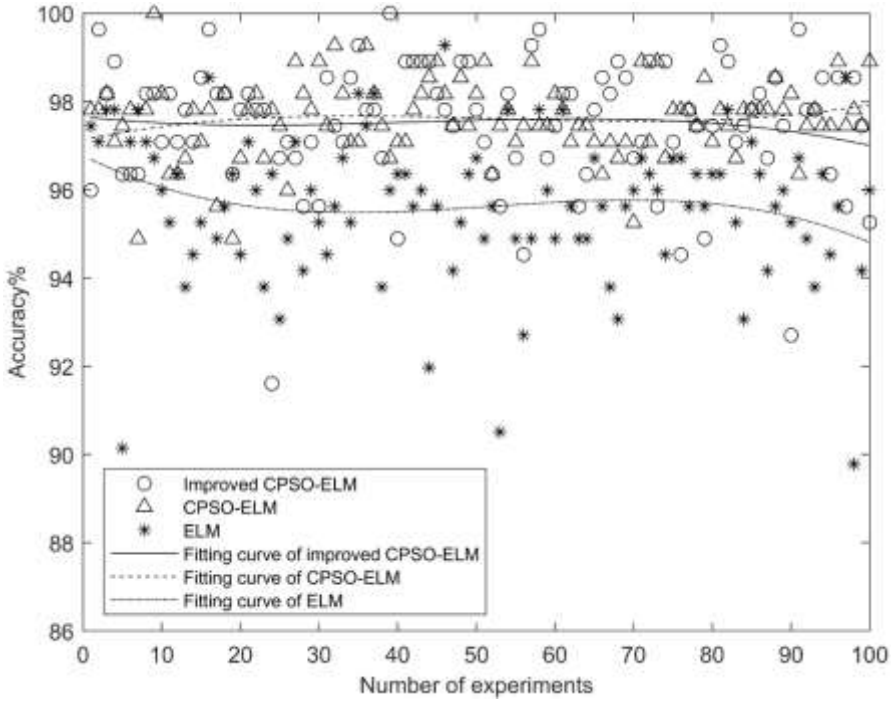

Figure 2 Accuracy results obtained by ELM, CPSO-ELM and improved CPSO-ELM

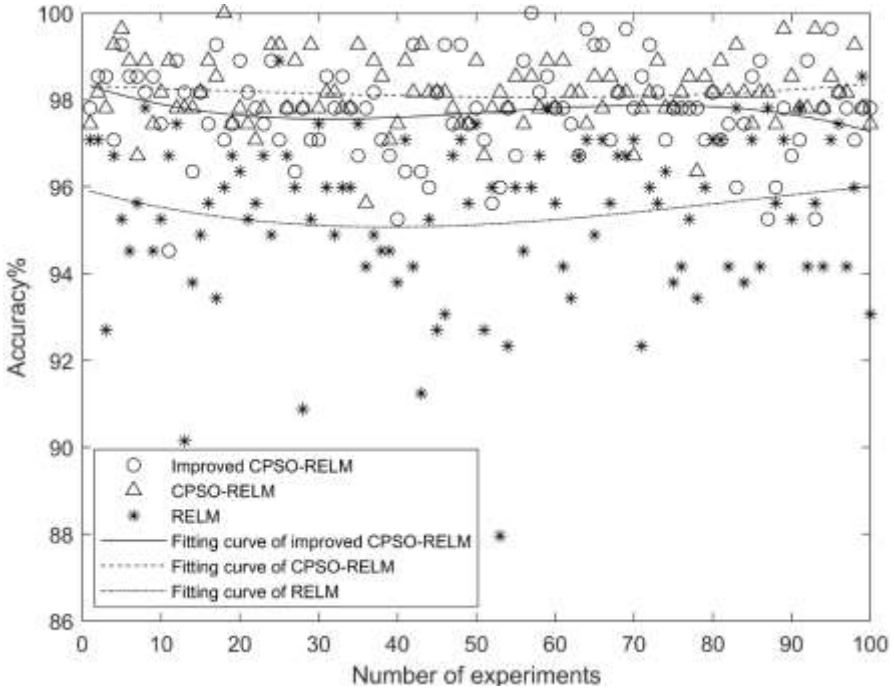

Figure 3 Accuracy results obtained by RELM, CPSO-RELM and improved CPSO-RELM

Table 1 Average accuracy results obtained by ELM, CPSO-ELM and improved CPSO-ELM

Condition	Average accuracy
ELM	95.65%
CPSO-ELM	97.59%
Improved CPSO-ELM	97.48%

Table 2 Average accuracy obtained by RELM, CPSO-RELM and improved CPSO-RELM

Condition	Average accuracy
RELM	95.41%
CPSO-RELM	98.16%
Improved CPSO-RELM	97.73%

We can see from the figures and tables above that CPSO-ELM/RELM and improved CPSO-ELM/RELM tend to have obvious higher accuracy and stability than ELM/RELM. However, for CPSO-ELM/RELM and improved CPSO-ELM/RELM, their accuracy have no obvious difference.

4.3 The Results of Running Time of ELM/RELM, CPSO-ELM/RELM and improved CPSO-ELM/RELM.

To prove that this method can promote running speed, we test running time of ELM/RELM, CPSO-ELM/RELM and improved CPSO-ELM/RELM. However, because the amount of data is too small, the running time will be effected to a large of extend. Therefore, we copy the data into 10 copies, which will be seen as the final testing data. We put it into the models and record running time. To ensure the precision of data, we do 100 experiments and use average value as the final results. The results are shown as follow:

Table 3 Running time of ELM, CPSO-ELM and improved CPSO-ELM

Algorithm	Number of hidden layer nodes	Running time(s)
ELM, CPSO-ELM	77	0.0468
Improved CPSO-ELM	13	0.0341

Table 4 Running time of RELM, CPSO-RELM and improved CPSO-RELM

Algorithm	Number of hidden layer nodes	Running time(s)
RELM, CPSO-RELM	324	0.0894
Improved CPSO-RELM	13	0.0342

The above table shows that the number of hidden layer nodes selected by CPSO algorithm is much more than by $L = \sqrt{n+m} + \omega$. Combining **4.2** for analysis, we can see that the accuracy of CPSO-ELM/RELM and improved CPSO-ELM/RELM have no obvious difference. But the running time of improved ELM/RELM is much less than CPSO-ELM/RELM, which means we can use this method to reduce running time and keep high precision in some fields that need timeliness.

5. Conclusions

In this paper, we propose a method to reduce the number of hidden layer nodes to get faster speed and keep high accuracy at the same time. ELM/RELM, CPSO-ELM/RELM and improved CPSO-ELM/RELM are used in the experiments. Through the simulation results, for ELM, we can see that the accuracy of CPSO-ELM is 1.94% more than ELM while it is 0.11% more than improved ELM. The speed of improved CPSO-ELM is 1.37 times faster than CPSO-ELM. And for RELM, the results show that the accuracy of CPSO-RELM is 2.75% more than RELM while it is 0.43% more than improved CPSO-RELM. The speed of improved CPSO-RELM is 2.61 times faster than CPSO-RELM. The above data indicate that for both ELM and RELM, this method can effectively promote the speed and reach high accuracy, which will make it more feasible to be used in some fields that require timeliness such as images recognition[13], Speech recognition

and Unmanned aerial vehicle driving technology.

6. Acknowlegements

This research is supported by Science and Technology Department of Sichuan Province, Fund of Science and Technology Planning (No. 2018JY0290), Meteorological Information and Signal Processing Key Laboratory of Sichuan Higher Education Institutes (No. QXXCSYS201606)

7. References

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

[2] Hu, F. , Wang, L. , Wang, S. , Liu, X. , & He, G. . (2016). A human body posture recognition algorithm based on bp neural network for wireless body area networks. *China Communications, 13*(8), 198-208.

[3] Argyriou, A., Baldassarre, L., Micchelli, C. A., & Pontil, M. (2013). On sparsity inducing regularization methods for machine learning. *Researchgate Net*, 205-216.

[4] Deng, W. , Zheng, Q. , & Chen, L. . (2009). Regularized extreme learning machine. *2009 IEEE SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE AND DATA MINING*, 389-395.

[5] Zhang, Y. B., & Xing, L. Z. (2016). Algorithm of speech endpoint detection based on wavelet analysis and pso-elm. *Journal of North University of China*.

[6] Chen, S., Yue, S., & Wu, M. (2016). Application of PSO-ELM in electronic system fault diagnosis. *IEEE International Conference on Prognostics & Health Management*.

[7] Liu, Y., Zhong, P., Zhang, M., Jin, G. U., & Kong, Y. (2013). Empirical formulae comparison for number selection of hidden layer's nodes in ann model applied for reservoir optimal operation. *Water Power*.

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

[9] Wang, S., Wang, Y. X., Shan, H. L., & Shi, C. X. (2018). Multimodal computer image recognition based on depth neural network. *Cluster Computing*(45), 1-7.

[10] Ye, Q. , Yuan, S. , & Kim, T. K. . (2016). Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation.

[11] Shi, Yuhui, Eberhart, & Russell, C. (1998). *Parameter Selection in Particle Swarm Optimization*. *Evolutionary Programming VII*.

[12] Tang, X., Ling, Z., Cai, J., & Li, C. (2010). Short communication: multi-fault classification based on support vector machine trained by chaos particle swarm optimization. *Knowledge-Based Systems, 23*(5), 486-490.

[13] Wang, S., Wang, Y. X., Shan, H. L., & Shi, C. X. (2018). Multimodal computer image recognition based on depth neural network. *Cluster Computing*(45), 1-7.