International Journal of Grid and Distributed Computing
Vol. 10, No. 1 (2017), pp.233-244
http://dx.doi.org/10.14257/ijgdc.2017.10.1.20

Selection ELM Parameters Based on Evolution of Modified Differential

Yibo Li, Yanghui Ren and Haixia Zhang

School of Automation
Shenyang Aerospace University
Shenyang 110136, Liaoning, China
liyibo_sau@163.com，Renhh_624@163.com

Abstract

The correct setting of hidden layer parameters are crucial to learning results and generalization ability of Extreme Learning Machine (ELM). In order to improve the ELM classification accuracy and generalization ability, the parameters which based on Modified Differential Evolution (MDE) algorithm is proposed. In this paper, parameters' selection is regarded as compound optimization problem and a MDE algorithm was proposed to select suitable parameters value. The experiment results demonstrate that the ELM based on MDE has better approximation performance and good generalization performance. Meanwhile, the MDE algorithm to optimize the parameters of hidden layer ELM selection which could improve the convergence rate of original differential evolution algorithm, accelerated the speed of ELM search parameters, and improved the ELM classification accuracy. Furthermore, it has the superior capability of function approximation, good global convergence ability and high optimization precision.

***Keywords**: Extreme Learning Machine; parameters selected; Modified Differential Evolution; function approximation*

1. Introduction

Extreme learning machine (ELM) have been investigated in Huang et al. It was easily wielded, effective single-hidden layer feed forward neural network [1]. It should be noted that the input weights (linking the input layer to the first hidden layer) and hidden layer biases needed to randomly allocated. Output layer weights directly calculated by the least squares method. The whole learning process once completed, no iteration, which can achieve fast learning speed. At present, parameter selection have no uniform standard and theory, people tended to depending on their experience or use cross-validation methods. Optimum parameters obtained by large number of tests. However, this method is time-consuming and the obtained parameters are not necessarily the best. Differential Evolution (DE) algorithm is proposed by Store and Price in 1995 as a novel intelligent optimization algorithm. It is based on groups heuristic search algorithm. Through cooperation and competition between individuals within populations to achieving the optimization problem solving. DE algorithm is a global optimization algorithm following the genetic algorithm. Which is conducive to maintaining the diversity. In addition to its fast convergence speed and good robust property, DE is a simple and efficient method for solving the global optimization problems. The input weights and hidden layer biases need to be adjusted in all these previous theoretical research works, as well as in almost all practical learning algorithms of feed forward neural networks.

In recent years, a number of parameters optimization methods was proposed. Ref [2] using gradient descent conduct parameter optimization. Although this method shortens the search time parameters but it requires a higher initial point selected. It is a linear search method, which can easily fall into local optimum. Ref [3] proposed genetic algorithm, Ref [4] proposed particle swarm optimization, Ref [5] proposed ant colony algorithm. They are separately make optimization selection for ELM parameters of hidden layer. These intelligent algorithms although reducing dependence on initial selection, and the algorithm principles and ideas more complex. Different optimization problems need to design different crossover, mutation and selection mode. Ref [6, 7] proposed differential evolution algorithm and applied it in parameter selection of support vector machine (SVM) or ELM. Accelerating the search speed parameters while improving the classification accuracy of SVM or ELM.

In order to improve method of the selection ELM parameters based on MDE algorithm, a optimizing the parameter of ELM hidden layer has been proposed. The optimizing criteria has been set under the rule which depends on minimizing the misjudgment rate, which a target function has been built. The optimal choice for the ELM input weights and thresholds (hidden layer bias) has been made through MDE algorithm in order to enhance the study ability and expand ELM's application. So as to improve the ELM learning capability and expanded it scope of application. The ELM based on MDE has better approximation performance and good generalization performance. Meanwhile, the MDE algorithm to optimizing the parameters of hidden layer ELM which could improve the convergence rate of original DE algorithm, accelerated the speed of ELM search parameters, and improved the ELM classification accuracy. Furthermore, it has the more excellent capability of function approximation, good global convergence ability and high optimization precision.

2. Extreme Learning Machine and Differential Evolution

2.1 Extreme Learning Machine

For N arbitrary distinct samples $(x_j, t_j) \in R^n \times R^m$, standard SLFNs with $\tilde{N}$ hidden neurons and activation function g(x) are mathematically modeled as [7]

$$\sum_{i=1}^{\tilde{N}} \beta_i g(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g(\omega_i x_j + b_i) = o_j \qquad (1)$$

Where $\omega_i = [\omega_{i1}, \omega_{i2}, \cdots \omega_{in}]^T$ is the weight vector connecting the *i*th hidden neuron and the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots \beta_{in}]^T$ is the weight vector connecting the *i*th hidden neuron and the output neurons, and *bi* is the threshold of the *i*th hidden neuron. $w_i \cdot x_j$ denotes the inner product of w_i and x_j.

The fact that standard SLFNs with $\tilde{N}$ hidden neurons each with activation function *g(x)* approximate these *N* samples errors can be zero, means that $\sum_{j=1}^{\tilde{N}} \| O_j - t_j \| = 0.i.e$,there exist β_i, w_i and b_i such that $\sum_{i=1}^{\tilde{N}} \beta_i g(\omega_i x_j + b_i) = \mathrm{t}_j$, $j = 1,2.\cdots, N$ .The above N equations can be written as: $H\beta = T$. where

$$H = \begin{Bmatrix} g(\omega_1 x_1 + b_1) & \cdots & g(\omega_{\tilde{N}} x_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\omega_1 x_N + b_1) & \cdots & g(\omega_{\tilde{N}} x_{\mathrm{N}} + b_{\tilde{N}}) \end{Bmatrix}_{N \times \tilde{N}}, \beta = \begin{Bmatrix} \beta_1^{\ T} \\ \vdots \\ \beta_{\tilde{N}}^{\ T} \end{Bmatrix}_{\tilde{N} \times m}, T = \begin{Bmatrix} t_1^{\ T} \\ \vdots \\ t_N^{\ T} \end{Bmatrix}_{N \times m}$$

H is called the hidden layer output matrix of the SLFN, the *i*th column of *H* is the *i*th hidden node output with respect to inputs is called the hidden layer feature mapping. The *i*th row of *H* is the hidden layer feature mapping with respect to the *i*th input. It has been proved [8] that from the interpolation capability point of view, if the activation function g is infinitely differentiable in any interval the hidden layer parameters can be randomly generated. The original ELM algorithm can be summarized as follows

Given the training set $\aleph = \left\{ (x_j, t_j) \middle| x_j \in R^n, t_j \in R^m, j = 1,2.\cdots, N \right\}$ and the activation function is *g(x)*, *N* is the number of hidden nodes.

1 Randomly selected input weights ω_i and the threshold b_i.

2 Calculation the hidden layer output matrix *H*.

3 Calculate the output weights *β*, $\beta = H^{+}T$ where H^{+} is the Moore-Penrose generalized inverse of matrix *H*.

2.2 Modified Differential Evolution

1) Basic Differential Evolution

Differential Evolution is a population-based evolutionary algorithm. Individuals with memory and optimal information sharing within population characteristics. That is the solution to the optimization problem through cooperation and competition between individuals within populations [9]. Firstly, set of random initialization of the population $X^0 = \left[x_1^0, x_2^0, \cdots, x_{N_p}^0 \right]$.

Np is the number of members in a population. It is not changed during the evolution process. The initial population is chosen randomly with uniform distribution in the search space. After a series of predetermined operation, the S-Generation individual evolved $x_i^s = \left[x_{i,1}^s, x_{i,2}^s, \cdots, x_{i,D}^s \right]$.

In the formulas, a set of *D* optimization parameters is called an individual. It is represent by a *D*-dimensional parameter vector.

DE has three operations: mutation, crossover and selection. Subtracted the parent of two different random individuals, obtaining the difference vector, then added to the first three randomly selected individuals. Generate a variation of the individual. Next, according to a certain probability placed crossover between the parents individuals, the individual variation will be generated a test subject. Then operated selection according to the size of the individual. Choose the better individual fitness as a child, to ensuring the evolution toward optimal direction [10].

a) Mutation

Using equation (4), for each target vector x_h^t, a mutant vector v_h^{t+1} is generated according to

$$v_i^{t+1} = x_{r_1}^t + F \cdot \left(x_{r_2}^t - x_{r_3}^t \right) \tag{2}$$

The r_1, r_2 and r_3 are randomly chosen indexes an $r_1, r_2, r_3 \in \left\{ 1, 2, \cdots, N_p \right\}$. Note that indexes must be different from each other. *F* is a real number to control the amplification of the difference vector $\left(x_{r_2}^t - x_{r_3}^t \right)$. According to Ref [6], the range of F is in $\left(0, 2 \right]$. If a component of a mutant vector goes off the search space, then this component is set to bound value.

b) Crossover

To increasing the diversity of the parameter vector, introduction of crossover operator. The target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector *u*.

$$u_{ij} = \begin{cases} v_{ij}^{t+1}, rand(j) \le CR \;\; or \;\; j = randn(i) \\ x_{ij}^{t}, rand(j) > CR \;\; and \;\; j \ne randn(i) \end{cases} \tag{3}$$

Where $j = 1,2,\ldots,D, rand(j) \in [0,1]$ is the j_{th} evaluation of a uniform random generator number, $CR \in [0,1]$ is the crossover probability constant, which has to be determined previously by the user. $randn(i) \in \{1,2,\cdots,D\}$ is a randomly chosen index which ensures that u_i gets at least one element from v_i^{t+1}. Additionally, the population have not new parents vector would be joined so it will not change.

c) Selection

DE adapts greedy selection strategy. If the trial vector u_i yields a better fitness function value than $\mathrm{x}_i^{\mathrm{t}}$, where u_i is set to x_i^{t+1}. Otherwise, the old value $\mathrm{x}_i^{\mathrm{t}}$ is retained. In this paper the minimization optimization is considered. The selection operator is as following.

$$x_i^{t+1} = \begin{cases} u_i^{t+1}, \; f(u_i^{t+1}) < f(x_i^t) \\ x_i^t, \; f(u_i^{t+1}) \ge f(x_i^t) \end{cases} \tag{4}$$

Among them, *J* as fitness function.

2) Time-varying Crossover Probability Strategy

From the crossover operator equation (3), we can see that with the increasing of iteration times the *CR* becomes larger accordingly from a littler value, then the $\mathrm{x}_i^{\mathrm{t}}$ contribute more to the u_i in the beginning stage and v_i^{t+1} has more contribution to the u_i in the later stage. As a result, the algorithm has good global exploration ability in the beginning stage and has good local searching ability in the later stage [11].

Various methods have been proposed in order to speed up the convergence rate and avoided premature convergence, this paper introduced a Time-varying crossover probability strategy. Supposed *g* was the current iteration and *G* was the maximum iteration times, the time-varying crossover probability constant *CR* is determined as the following equation:

$$\mathrm{CR} = CR_{\min} + \frac{g(CR_{\max} - CR_{\min})}{G} \tag{5}$$

Where the *CR* $_{\min}$, *CR* $_{\max}$ are the minimum crossover probability constant and the maximum crossover probability constant respectively. So crossover probability factor with the iterative algorithm proceeds linearly increases. *CR* too large, in favor of local search and accelerate the convergence rate. On the contrary, *CR* is conducive to maintaining population diversity and conduct global search. Good search strategy should be to keep the population diversity and global search in the initial stage of the search. In the latter part of the search should strengthen the local search capability to improve the accuracy of the algorithm. In the search process, dynamically change the value of *CR* according to population diversity.

Root causes of premature convergence is the iteration count increases and the rapid decline of population diversity, there has been a 'gathering'. In order to quantitatively describe the state of the population, the following definitions are given group fitness variance.

Definition 1 Let population size is N_p, *fi* is the *i*th individual fitness, f_{avg} is the current average fitness of the population. So δ^2 can be defined as:

$$\delta^2 = \sum_{i=0}^{N_p} \left| \frac{f_i - f_{avg}}{f} \right|^2 \quad (6)$$

Where f is the normalized scaling factor, its role is limited to the size of δ^2, whose value is

$$f = \begin{cases} \max\{|f_i - f_{avg}|\}, & \max\{|f_i - f_{avg}|\} > 1 \\ 1 \ , & other \end{cases} \quad (7)$$

Definition 1 shows that group fitness variance δ^2 reflecting the degree of aggregation of all individuals in the population. On the contrary, populations are in random search stage. During the DE algorithm, if the group fitness variance equal to zero, the optimal solution is not theoretical optimal solution obtained at this time. The population into a local optimum and the algorithm premature convergence.

3. MDE Algorithm of ELM Parameter Optimization

Traditional ELM network structure set by the user experience, and needed too many times experiment .The test results in order to selecting the appropriate network architecture as the ultimate network model for training and testing. This method is complicated and time-consuming, increasing artificial workload and needing to constantly adjust the network structure replicates. Presented an optimization method of ELM--Extreme Learning Machine based on the Modified Differential Evolution.

For different problems, MDE_ELM algorithm through improved differential evolution of global optimization capabilities, made the input weights and thresholds (hidden layer offset) to be more effective. And improved the ELM hidden layer node effectiveness. Thereby enhancing the learning capacity of the whole and extended the applicable scope of ELM. Optimized network structure made ELM effectively adapt to different problems with least human intervene.

3.1 Network Structure of ELM

ELM structural factors mainly refers to the number of network nodes hidden layer affect the ability of generalization. Research has shown that the number of hidden layer nodes too little can limited ability to learn, and cause learning task cannot be completed. Excessive can lead to "over-fitting" phenomenon, and reduce the generalization ability of the network. Met the accuracy requirements, less of nodes would be better when users design ELM network structure. ELM network optimization evolutionary algorithm is mainly dependent on the fitness function to guide the search strategy. Therefore, the definition of the fitness function is particularly important.

3.2 Fitness Function

Evolutionary algorithms are mainly dependent on the fitness function to guide the search strategy. Therefore, the definition of the fitness function is particularly important. Optimization objective function defined here is individual fitness function.

$$\min f = k\left(\lambda_1 \frac{m_1}{n_1} + \lambda_2 \frac{m_2}{n_2} \right) \quad (8)$$

Where m_1, m_2 represent the number of samples types of false positives. n_1, n_2 represent two types of the number of samples. $\frac{m_1}{n_1}, \frac{m_2}{n_2}$ are misdiagnosis rate of category 1 and misdiagnosis rate of category 2. k is a scaling factor. Can be used to control variety of the objective function value. $\lambda_1, \lambda_2 \in [0,1]$ are misjudgment categories of the control factor. Can be controlled by the size of λ_1, λ_2.

3.3 Optimization procedures of MDE

The searching procedures of the Modified Differential Evolution (MDE) were shown as below. The maximum number of generations through many experiments take an appropriate value.

Step1: Specify the number of population N_P, the difference vector scale factor F, the minimum and maximum crossover probability constant $CR_{\min}$ and $CR_{\max}$, and the maximum number of fitness evaluation times. Initialize randomly the individuals of the population and the trial vector in the given searching space.

Step2: Calculate the fitness value of each individual in the population using the objective function given by equation (8).

Step3: Compare each determine whether the individual fitness reaches a predetermined accuracy or Satisfy that $g=g_m$. If meet which term return to Step9. Otherwise, do the next step.

Step4: Compare each individual's fitness value and get the best fitness and best individual.

Step5: Generate a mutant vector according to equation (2) for each individual.

Step6: According to equation (3), do the crossover operation and yield a trial vector.

Step7: Do the selection operation in terms of equation (4) and generate a new population.

Step8: $t=t+1$, return to Step3 until to the maximum number of generations.

Step9：Get the optimized parameters. Use the sample data for training and testing. On this account, conduct regression and classification.

4. Simulation Experiment

In order to verify the optimizing ability of the MDE-ELM. Respectively simulation in both regression and classification of the function. The experimental data from the UCI standard database. The selected experimental data sets are described on below Table 1:

Table 1. Specifications of Data Sets

Data sets	Attributes	Category	Training data	Testing data
Diabetes	8	2	576	192
Banana	2	2	800	400
Sonar	60	2	80	114
Glass	9	6	142	72
Iris	4	3	100	50
Segment	18	7	1000	800

4.1 Regression Experiment

Root mean square error (RMSE) is the least mean square of the actual output and the predicted output. It used to represent the network of prediction accuracy.

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\left(O_i - t_i\right)^2} \tag{9}$$

Where N is the number of the training samples. t_i is ideal output value of the ith sample. O_i is actual output value of the ith sample. It is a fitting application evaluation of the represents ELM fits the data set on the predictive capability. RMSE becomes smaller, the ELM learning and predictive ability becomes stronger. Otherwise, the ELM learning and predictive ability becomes weaker.

In this example, all the four algorithms (ELM, DE_ELM, PSO_ELM and MDE_ELM) are used to approximate the '*SinC*' function.

$$f(x) = \begin{cases} \dfrac{\sin x}{x} & x \neq 0 \\ 1 & x = 0 \end{cases} \tag{10}$$

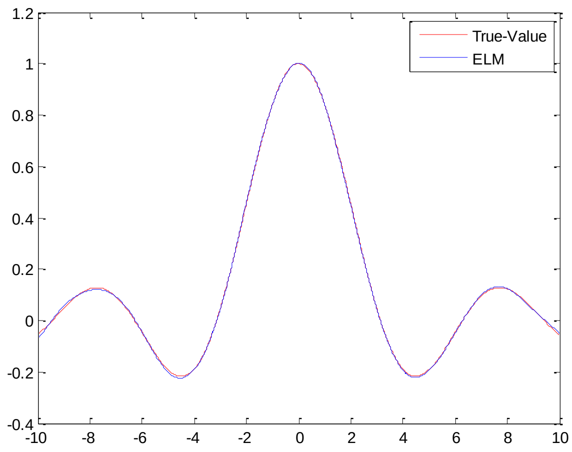

Figure 1. The image of Sine Function in the [-10, 10] Range

In order to make the regression problem 'real', large uniform noise distributed in $[-0.2,0.2]$has been added to all the training samples while testing data remain noise-free. A training set $\{x_i, f(x_i)+\zeta_i\}$and testing set $\{x_i, f(x_i)\}$with 5000 data, respectively, are created where xi are uniformly randomly distributed on the interval $[-10,10]$.

Table 2. Function Result of 'Sinc' under the Same Network Structure

Algorithms	parameters	Iterations	Training Error	Testing Error	Testing Time (s)
ELM	N_h =100		0.1180	0.0282	0.00940
DE_ELM	N_P =200 N_h =100 CR=0.8	20 50	0.1156 0.1158	0.0131 0.0142	15.0918 18.7215
PSO_ELM	N_P =200 Nh =100	20 50	0.1156 0.1200	0.0122 0.0143	14.8575 16.2421
MDE_ELM	N_P =200 N_h =100 CR=0.8	20 50	0.1158 0.1159	0.0078 0.0080	5.0301 9.0654

Where N_P is the number of population, N is iterations, crossover probability constant is *CR*. Observed the training error and testing error ,it can be see that using swarm intelligence algorithm to obtain input weights and thresholds can improved the effectiveness of the hidden layer nodes. Thereby MDE algorithm improves the learning performance of ELM.

4.2 Classification Experiment

Classification accuracy of the training set (CATS)

$$CATS = 1 - \frac{r}{n} \tag{11}$$

Extreme Learning Machine's main application is the application in the field of pattern recognition. It can construct a classifier. Where r denotes the test set number of samples misclassified. *n* represents the total number of test set. It represents the ELM on the proper classification of data capacity. It is a common classification of evaluation. The higher classification accuracy make the learning and predicting ability better. Otherwise be a worse and lower.

Table 3. The Experimental Results of Two-category

Data sets	Algorithms	Hidden layer nodes	Training Rate （%）	Testing Rate （%）	Testing Time (s)
Diabetes	ELM	50	73.91	71.67	0.0031
	DE_ELM	50	76.30	76.04	8.7095
	PSO_ELM	50	76.12	75.07	9.2546
	MDE_ELM	50	**83.68**	**83.33**	8.1652
Banana	ELM	100	77.00	79.05	0.0562
	DE_ELM	100	84.74	81.96	9.0575
	PSO_ELM	100	84.65	85.91	10.1350
	MDE_ELM	100	**90.75**	**92.00**	7.0822
Sonar	ELM	100	74.65	73.12	0.0092
	DE_ELM	100	85.13	81.62	8.5975
	PSO_ELM	100	83.81	84.17	11.9824
	MDE_ELM	100	**91.62**	**92.51**	9.8617

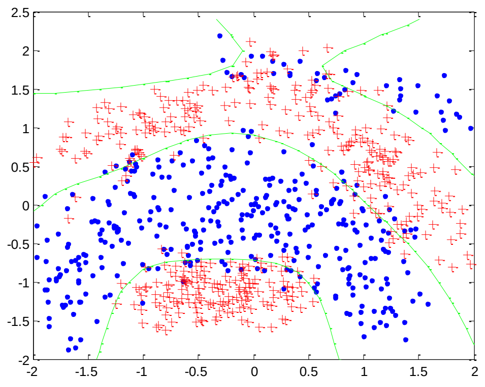

Figure 2. ELM Algorithm

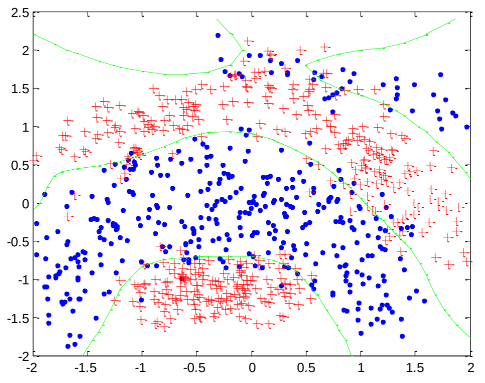

Figure 3. PSO_ELM Algorithm

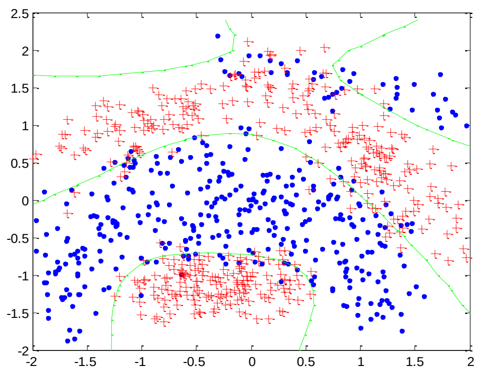

Figure 4. DE_ELM Algorithm

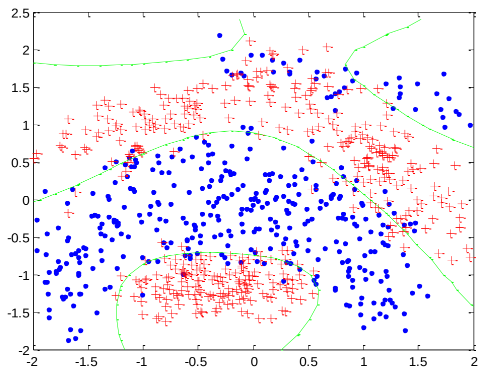

Figure 5. MDE_ELM Algorithm

The Table 4 is the statistical results of the comparison. The Figure 2 to Figure 5 illustrates the test results of these four algorithms in Banana data-sets categories renderings. From these simulations, it can be seen that ELM achieves shorter training time than DE_ELM, PSO_ELM and MDE_ELM. MDE_ELM in Banana data-sets categories renderings classification was better than the other three algorithms. Because introduced the swarm intelligence algorithm into ELM input weights and thresholds in the selection process takes a lot of time for intelligent optimization. DE_ELM, PSO_ELM and MDE_ELM in the iterative optimization process, consuming almost the same time. The four algorithms in Diabetes, Banana and Sonar, the training accuracy and testing accuracy of MDE_ELM will be higher than the ELM, DE_ELM and PSO_ELM. Therefore, MDE_ELM algorithm is an efficient algorithm for evolution in binary classification applications.

Table 5. The Experimental Results of Multi-classification

Data sets	Algorithms	Hidden layer nodes	Training Rate （%）	Testing Rate （%）	Testing Time (s)
Segment	ELM	100	79.39	78.47	0.07200
	DE_ELM	100	87.53	80.99	29.6773
	PSO_ELM	100	88.96	81.68	30.4901
	MDE_ELM	100	**92.79**	**84.68**	28.5110
Glass	ELM	100	71.83	70.83	0.0312
	DE_ELM	100	78.87	69.44	5.9483
	PSO_ELM	100	88.62	86.24	9.1458
	MDE_ELM	100	**90.43**	**89.72**	7.4625
Iris	ELM	100	92.00	88.65	0.7802
	DE_ELM	100	91.25	90.04	3.2458
	PSO_ELM	100	93.06	92.15	7.1357
	MDE_ELM	100	**94.00**	**92.15**	5.2167

As can be seen from the table 5. In the case of having the same training accuracy with other algorithms, MDE_ELM algorithm can be obtained higher testing accuracy. So MDE_ELM algorithm can be accelerated the response speed of network location data and effectively improved testing accuracy.

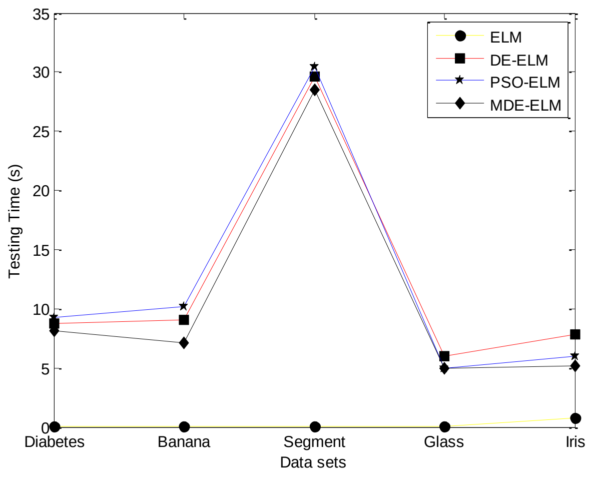

Figure 6. The Test Time of Four Different Algorithms on Data-set

Seen from Figure 6, obviously the MDE_ELM algorithm testing time on the same data set to be smaller than the other two swarm intelligence algorithms.

Therefore, intelligent algorithm into ELM input weights and threshold selection process can improve the performance of ELM.

In this paper, an effective algorithm be proposed. Simulations results show that the ELM whose parameters selected by MDE has better performance in multi-classification application.

5. Conclusions

As a learning technique, MDE_ELM has demonstrated good potentials to resolving regression and classification problems [12]. Since the traditional ELM random selection input weights and thresholds, so leads to complex problems in network architecture. Taking into account, improved DE algorithm have advantage of strong optimization ability and less control parameters, etc. [13]. This work proposed an extreme learning machine based on modified differential algorithm. Introduced the MDE algorithm into selection process of input weights and thresholds. It can get a more appropriate input weights and thresholds. Meanwhile, the MDE algorithm to optimize the parameters of hidden layer ELM selection which could improve the convergence rate of original differential evolution algorithm, accelerated the speed of ELM search parameters, and improved the ELM classification accuracy. Making more effective the hidden layer nodes to improved performance on ELM.

Through the designed and simulation experiment analysis the MDE_ELM, its ability of learning and forecasting capabilities are better than traditional ELM. It can be more streamlined network architecture to achieve the same or better learning and predictive capability. This algorithm accelerated the network respond rate of the unknown data. It will be an effective way that putting ELM into practice where requires response time instead of training time. Therefore, MDE_ELM algorithm is an effective algorithm of combined with ELM and intelligent algorithm. In view of this, the combination of these two methods may result in widely applications in the neural networks area.

Acknowledgments

The project supported by the Program for Liaoning Excellent Talents in University (No. LJQ2014018). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: a new learning scheme of feed forward neural networks ", Neurocomputing, vol. 2, no. 2, **(2004)**, pp. 985-990.
[2] O. Chappelle, V. Vapnik and O. Bousquet, "Choosing multiple parameters for support vector machines", Machine Learning, vol. 46, no. 1, **(2002)**, pp. 131-160.
[3] Z. Chunhong and J. Licheng, "Automatic parameters selection for SVM based on GA", Proceedings of the 5th World Congress on Intelligent Control and Automation, IEEE Press, **(2004);** Piscataway, NJ.
[4] S. Xinguang, Y. Huizhong and C. Gang, "Parameters selection and application of support vector machines based on particle swarm optimization algorithm", Control Theory and Applications, vol. 23, no. 5, **(2006)**, pp. 740-743.
[5] Q. Liang, "Parameters Selection of Support Vector Machine Based on Ant Colony Algorithm", System Simulation Technology, vol. 11, **(2008)**, pp. 34-38.
[6] C. Tao, Y. Longquan and D. Fang' an, "Parameters selection of support vector machine based on differential evolution", Computer Engineering and Applications, vol. 47, no. 5, **(2011)**, pp. 24-26.
[7] Y. Zhang, B. Liu and J. Cai J, "Ensemble weighted extreme learning machine for imbalanced data classification based on differential evolution", Neural Computing & Applications, **(2016)**, pp. 1-9.
[8] Z. Min, Z. Rui and W. Yu, "A Summary of Network Structure Adjustments of Extreme Learning Machine", Journal of Xi, an University of Arts ＆ Science(Nat Sci Ed), vol. 17, no. 1, **(2014)**, pp. 1-6.
[9] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: theory and applications". Neurocomputing , vol. 70, **(2006)**, pp. 489–501
[10] J. W. Zhao, Z. H. Wang and D. Park, "Online sequential extreme learning machine with forgetting mechanism", Neural Computation, vol. 87, **(2012)**, pp. 79-89.
[11] P. K. Wong, C. M. Vong, X. H. GAO and K. I. Wong, "Adaptive Control Using Fully Online Sequential-Extreme Learning Machine and a Case Study on Engine Air-Fuel Ratio Regulation", Hindawi Publishing Corporation Mathematical Problems in Engineering, **(2014)**.
[12] S. W. Zhou, L. H. Wu, X. F. Yuan and W. Tan, "Parameters Selection for Function Approximation Based on Differential Evolution", International Journal of Computational Intelligence Systems, **(2007)**, p. 10.
[13] G.-B. Huang, D.-H. Wang and Y. Lan, "Extreme learning machines: a survey", Springer-Verlag Int. J. Mach. Learn. & Cyber, vol. 2, **(2011)**, pp. 107–122.
[14] S. Scardapane, D. Comminiello, M. Scarpiniti and A. Uncini, "Online Sequential Extreme Learning Machine with Kernels", Neural Networks, IEEE Transactions, vol. 26, no. 9, **(2015)**, pp. 2214-2220.
[15] G.-H. Li, M. Liu and M. Y. Dong, "A new online learning algorithm for structure-adjustable extreme learning machine", Computers and Mathematics with Applications, vol. 60, **(2010)**, pp. 377-389.
[16] G. Zhang, X. S. Xie, Y. Huang and C. R. Wang, "An online multi-kernel learning algorithm for big data", CAAI Transactions on Intelligent Systems, vol. 9, no. 3, **(2014)**.

Authors

Yibo Li, he was born in 1963, he received master degree from Nanjing University of Aeronautics and Astronautics in 1988 and doctor degree from Northeastern University in 2003. He became a teacher in Shenyang Aerospace University since 1988. His main research interests are image processing, pattern recognition, flight control, etc.

Yanghui Ren, she was born in 1991, she received bachelor degree from Qingdao University of Technology in 2014. Her main research interests are computational intelligence pattern recognition and artificial intelligence.

Haixia Zhang, she was born in 1988, she received bachelor degree from Qingdao Agricultural University in 2014. Her main research interests are machine learning, pattern recognition and artificial intelligence.