Journal of System Simulation

Volume 29 | Issue 10 Article 28

6-4-2020

Evolutionary Extreme Learning Machine Optimized by Quantum-behaved Particle Swarm Optimization

Pang Shan
1. College of Information and Electrical Engineering, Ludong University, Yantai 264025, China;;

Xinyi Yang
2. Department of Aircraft Engineering, Naval Aeronautical and Astronautical University, Yantai 264001, China;

Xuesen Lin
2. Department of Aircraft Engineering, Naval Aeronautical and Astronautical University, Yantai 264001, China;

Evolutionary Extreme Learning Machine Optimized by Quantum-behaved Particle Swarm Optimization

Abstract

Abstract: Extreme Learning Machine (ELM) is a novel learning algorithm for Single-Hidden-Layer Feed Forward Neural Networks (SLFN) with much faster learning speed and better generalization performance than traditional gradient-based learning algorithms. However ELM tends to require more neurons in the hidden layer and lead to ill-conditioned problem due to the random selection of input weights and hidden biases. To address these problems, *a learning algorithm was proposed which used quantum-behaved particle swarm optimization (QPSO) to select the optimal network parameters including the number of hidden layer neurons according to the both the root mean square error on validation data set and the norm of output weights.* Experimental results on benchmark regression and classification problems have verified the performance and effectiveness of the proposed approach.

Keywords

extreme learning machine, single-hidden-layer feed forward neural networks, quantum-behaved particle swarm optimization, generalization performance

第 29 卷第 10 期 系统仿真学报© Vol. 29 No. 10
2017 年 10 月 Journal of System Simulation Oct., 2017

Evolutionary Extreme Learning Machine Optimized by Quantum-behaved Particle Swarm Optimization

Pang Shan[1], *Yang Xinyi*[2], *Lin Xuesen*[2]

(1. College of Information and Electrical Engineering, Ludong University, Yantai 264025, China;
2. Department of Aircraft Engineering, Naval Aeronautical and Astronautical University, Yantai 264001, China)

Abstract: Extreme Learning Machine (ELM) is a novel learning algorithm for Single-Hidden-Layer Feed Forward Neural Networks (SLFN) with much faster learning speed and better generalization performance than traditional gradient-based learning algorithms. However ELM tends to require more neurons in the hidden layer and lead to ill-conditioned problem due to the random selection of input weights and hidden biases. To address these problems, *a learning algorithm was proposed which used quantum-behaved particle swarm optimization (QPSO) to select the optimal network parameters including the number of hidden layer neurons according to the both the root mean square error on validation data set and the norm of output weights.* Experimental results on benchmark regression and classification problems have verified the performance and effectiveness of the proposed approach.

Keywords: extreme learning machine; single-hidden-layer feed forward neural networks; quantum-behaved particle swarm optimization; generalization performance

一种基于量子粒子群优化的极限学习机

逄珊[1]，杨欣毅[2]，林学森[2]

(1. 鲁东大学信息与电气工程学院，烟台 264025；2. 海军航空工程学院飞行器工程系，烟台 264001)

摘要：极限学习机(ELM)是一种新型的单隐含层神经网络的训练方法，同传统的基于梯度的网络训练方法相比，具有快速的学习速度和更好的泛化性能。ELM 在实际应用中往往需要大量的隐含层神经元，由于随机设定输入权值和偏置值，容易导致病态问题的出现。*为解决上述问题，提出一种应用量子粒子群(QPSO)优化包括隐含层节点个数在内的网络参数的方法。这种优化基于验证集的均方根误差，考虑到了输入权值矩阵的范数。*在典型的回归和分类问题上进行试验证明了算法的有效性。

关键词：极限学习机；单隐含层前馈神经网络；量子粒子群；泛化能力

中图分类号：TP183 文献标识码：A 文章编号：1004-731X (2017) 10-2447-12

DOI: 10.16182/j.issn1004731x.joss.201710028

Introduction

Extreme learning machine is a novel learning algorithm for single-hidden-layer feedforward neural networks proposed by Huang, et al[1-2]. In ELM, the input weights and hidden biases are randomly generated, and the output weights are calculated by Moore-Penrose (MP) generalized inverse. ELM learns much faster with higher generalization performance than the traditional gradient-based

Received: 2015-10-08 Revised :2015-12-15;
Foundation item: National Natural Science Foundation of China (61602229), Natural Science Foundation of Shandong Province (ZR2016FQ19);
Biography: Pang shan(1981-), female, Yantai, Shandong, master degree, lecturer, Research direction for include intelligent computation, theories and applications of machine learning.

learning algorithms such as back-propagation and Levenberg-Marquardt method. Also, ELM avoids many problems faced by traditional gradient-based learning algorithms such as stopping criteria, learning rate and local minima problem. However ELM may require more hidden neurons than traditional gradient-based learning algorithms and lead to ill-conditioned problem because the input weights and hidden biases are randomly selected.

In order to solve these problems, some Optimization methods have been combined with ELM for the training of SLFN. In Ref.[3], Zhu proposed an evolutionary ELM (E-ELM) which incorporates a widely used global searching method, differential evolution (DE) to optimize the input weights. Results show that the evolutionary ELM achieves good generalization performance with more compact networks.

In Ref.[4] a self-adaptive E-ELM (SaE-ELM) which also uses DE to optimize the network parameters is proposed. In this methodology the generation strategies and control parameters of the DE method are self-adapted by the optimization method. SaE-ELM outperforms E-ELM in most test cases.

In Ref.[5], a new learning framework called optimized extreme learning machine (O-ELM) is proposed. It uses optimization method to select not only the input weights, the hidden biases, the regularization factor but also the number of hidden neurons, thus the network structure is also optimized. Three optimization methods (GA, DE and Simulated Annealing) are tested in the framework on some benchmark regression problems and GA performs better than DE and SA.

Recently, Particle Swarm Optimization (PSO) has also been combined with ELM. PSO is a population based stochastic optimization technique. Compared with GA and DE, PSO has no complicated evolutionary operators such as crossover and mutation, and is easy to execute. Many researches proved that PSO shows better performance in complex optimization problems[6-8].

In Ref. [9], an evolutionary ELM optimized by PSO called PSO-ELM was proposed, and was applied in a prediction task. In Ref. [10], an Improved Extreme Learning Machine, IPSO-ELM, which uses an improved PSO to select the input weights and hidden biases of the SLFN was proposed. The IPSO-ELM optimizes the input weights and the hidden biases according to the RMSE on the validation data set and the norm of the output weights. Thus, IPSO-ELM algorithm can obtain good performance with more compact and well-conditioned SLFN than E-ELM and PSO-ELMs. However, the structure of SLFN is not optimized in IPSO-ELM, the number hidden neurons needs to be predefined by trials.

Although PSO outperforms GA and other evolutionary algorithms in many applications, it also has some limitations. PSO is not a global optimization algorithm and has premature or local convergence problems, as has been demonstrated in Ref. [11]. To solve these problems, Sun et al. proposed a quantum-behaved PSO (QPSO) algorithm[12-14]. Experiments show that QPSO shows better convergence performance than standard PSO, GA and some other algorithms in solving typical benchmark optimization problems[15-16].

The optimization of SLFN parameters is a very tough task as there are too many parameters to be optimized simultaneously. In order to optimize the network parameters more effectively, in this paper, a novel hybrid learning algorithm called QPSO-ELM

https://dc-china-simulation.researchcommons.org/journal/vol29/iss10/28
DOI: 10.16182/j.issn1004731x.joss.201710028

which takes advantage of the global search performance of QPSO is proposed. In the proposed algorithm, QPSO is used to search not only the optimal input weights and hidden biases, but also the best network structure. Therefore, we can achieve a more compact network without trial and error on the best number of hidden neurons. Furthermore, on the selection criteria of the optimization algorithms, we consider both the fitness value on validation data set and the norm of output weights to improve generalization performance of the network. The proposed algorithm was evaluated on both real-world regression and classification problems.

The rest of the paper is organized as follows. Section 2 gives a brief review of ELM. QPSO is overviewed in Section 3. The proposed QPSO-ELM is presented in Section 4. Section 5 gives the experimental results and discussion on both regression and classification problems. Finally concluding remarks are drawn in Section 6.

1 Brief of extreme learning machine

Extreme learning machine (ELM) was proposed by Huang, et al[1]. For N arbitrary distinct samples $(\boldsymbol{x}_i, \boldsymbol{t}_i)$, where $\boldsymbol{x}_i = [x_{i1}, x_{i2} \cdots, x_{in}]^T \in \boldsymbol{R}^n$ and $\boldsymbol{t}_i = [t_{i1}, x_{i2} \cdots, t_{im}]^T \in \boldsymbol{R}^m$. Standard SLFN with K hidden neurons and activation function $g(x)$ can approximate these N samples with zero error which means that

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{T} \tag{1}$$

where $\boldsymbol{H} = \{h_{ij}\}, (i = 1, 2, \cdots, N \text{ and } j = 1, 2, \cdots, K)$ is the hidden layer output matrix, $h_{ij} = g(\boldsymbol{w}_j \cdot \boldsymbol{x}_i + b_j)$ denotes the output of j-th hidden neuron with respect to $\boldsymbol{x}_i$, $\boldsymbol{w}_j = [w_{j1}, w_{j2}, \cdots, w_{jn}]^T$ is the weight connecting j-th hidden neuron and input neurons. b_j denotes the bias of j-th hidden neuron. And $\boldsymbol{w}_j \cdot \boldsymbol{x}_i$ is the inner product of $\boldsymbol{w}_j$ and $\boldsymbol{x}_i$. $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_K]^T$ is the matrix of output weights and $\boldsymbol{\beta}_j = [\beta_{j1}, \beta_{j2}, \cdots, \beta_{jm}]^T$ $(j = 1, \cdots, K)$ is the weight vector connecting the j-th hidden neuron and output neurons. And $\boldsymbol{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \cdots, \boldsymbol{t}_N]^T$ is the matrix of desired output.

Therefore, the determination of the output weights is to find the least-square (LS) solutions to the given linear system. The minimum norm LS solution to the linear system (1) is

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^+\boldsymbol{T} \tag{2}$$

where $\boldsymbol{H}^+$ is the MP generalized inverse of matrix H. The minimum norm LS solution is unique and has the smallest norm among all the LS solutions. ELM uses MP inverse method to obtain good generalization performance with dramatically increased learning speed.

2 Brief of quantum-behaved particle swarm optimization

QPSO is a novel optimization algorithm inspired by the fundamental theory of particle swarm and features of quantum mechanics. It was initially developed to deal with PSO's main limitation of pre-mature convergence. In QPSO, the state of a particle $\boldsymbol{y}$ is depicted by Schrodinger wave function $\psi(\boldsymbol{y}, t)$, instead of position and velocity. The dynamic behavior of the particle is widely divergent from classical PSO systems in that the exact values of position and velocity cannot be determined simultaneously. It can be only learned that the probability of the particle's appearing in a position from probability density function $|\psi(\boldsymbol{y}, t)|^2$, the form of which depends on the potential field the particle lies in. Employing the Monte Carlo method, for the i-th particle $\boldsymbol{y_i}$ from the population, the particle moves according to the following iterative equation:

$$\boldsymbol{y}_{i,j}(t+1)=\boldsymbol{P}_{i,j}(t)-\beta.(\boldsymbol{mBest}_j(t)-\boldsymbol{y}_{i,j}(t)).\ln(1/u_{i,j}(t))\ \ \textit{if } \boldsymbol{k}\geqslant \boldsymbol{0.5}$$
$$\boldsymbol{y}_{i,j}(t+1)=\boldsymbol{P}_{i,j}(t)+\beta.(\boldsymbol{mBest}_j(t)-\boldsymbol{y}_{i,j}(t)).\ln(1/u_{i,j}(t))\ \ \textit{if } \boldsymbol{k}< \boldsymbol{0.5} \quad (3)$$

where $\boldsymbol{y}_{i,j}(t+1)$ is the position of the i-th particle with respect to the j-th dimension in iteration t. $\boldsymbol{P}_{i,j}$ is the local attractor of i-th particle to the j-th dimension and is defined as

$$\boldsymbol{P}_{i,j}(t)=\varphi_j(t).\boldsymbol{pBest}_{i,j}(t)+(1-\varphi_j(t))\boldsymbol{gBest}_j(t) \quad (4)$$

$$\boldsymbol{mBest}_j(t)=\frac{1}{NP}\sum_{i=1}^{NP}\boldsymbol{pBest}_{i,j}(t) \quad (5)$$

where NP is the number of particles, $\boldsymbol{pBest}_i$ represents the best previous position of the i-th particle. $\boldsymbol{gBest}$ is the global best position of the particle swarm. $\boldsymbol{mBest}$ is the mean best position defined as the mean of all the best positions of the population, k, u and φ are random number distributed uniformly in [0, 1] respectively. β is called Contraction-Expansion coefficient. It can be tuned to control the convergence speed of the algorithms.

3 QPSO-ELM

In ELM, the output weights are computed based on random input weights and hidden biases, there may exist a set of non-optimal or even unnecessary input weights and hidden neurons. As a result, ELM may need more hidden neurons than conventional gradient based learning algorithms and lead to an ill-conditioned hidden output matrix, which would cause worse generalization performance.

In this section, a new approach named QPSO-ELM which combines QPSO and ELM is proposed. Unlike some other evolutionary ELMs, our proposed algorithm optimizes not only the input weights and hidden biases using QPSO, but also the structure of the neural network (hidden layer neurons). And this helps the algorithm to achieve a more compact network. The detailed steps of the proposed approach are as follows:

Step 1 Initializing: Firstly, we generate the population randomly. Each particle in the population is constituted by a set of input weights, hidden biases and s-variables.

$$\boldsymbol{p}_i=[w_{11},\cdots,w_{NK},b_1,\cdots,b_K,\cdots,s_1,\cdots,s_K]$$

Where $s_i\ i=1,\cdots,h$ is a variable which defines the structure of the network. As illustrated in Figure 1, if $s_i=0$, then the i-th hidden neuron is not considered. Otherwise, if $s_i=1$, the i-th hidden neuron is retained and the sigmoid function is used as its activation function.

All components in a particle are randomly initialized within the range [0, 1].

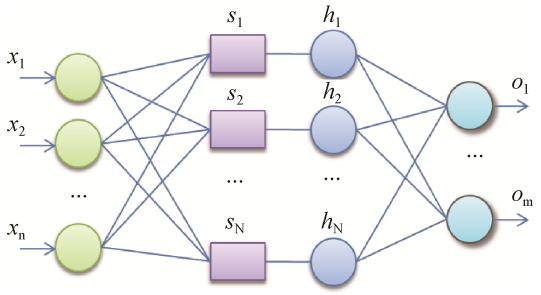


Fig. 1 Single hidden-layer feedforward network with s-variable

Step 2 Fitness evaluation: The corresponding output weights of each particle are computed according to Equation (6). Then the fitness of each particle is evaluated by the root mean square error between the desired output and estimated output. To avoid the problem of overfitting, the fitness evaluation is performed on the validation data set instead of the whole training data set

$$f(\ )=\sqrt{\frac{\sum_{j=1}^{N_v}\sum_{i=1}^{K}\boldsymbol{\beta}_i g(\boldsymbol{w}_i\cdot\boldsymbol{w}_j+b_i)-\boldsymbol{t}_j}{N_v}} \quad (6)$$

where N_v is the number of samples in the

https://dc-china-simulation.researchcommons.org/journal/vol29/iss10/28
DOI: 10.16182/j.issn1004731x.joss.201710028

validation data set.

Step 3 Updating $\boldsymbol{pBest}_i$ and $\boldsymbol{gBest}$: with the fitness values of all particles in population, the best previous position for i-th particle $\boldsymbol{pBest}_i$ and the global best position $\boldsymbol{gBest}$ of each particle is updated. However, using fitness value alone as the selection criteria is not enough. As suggested in Ref. [3], neural network tends to have better generalization performance with the weights of smaller norm. In this paper, the fitness value along with the norm of output weights are considered together for updating $\boldsymbol{pBest}_i$ and $\boldsymbol{gBest}$ .The updating strategy is as follows:

$$\boldsymbol{pBest}_i = \begin{cases} \boldsymbol{p}_i & (f(\boldsymbol{pBest}_i) - f(\boldsymbol{p}_i) > \eta f(\boldsymbol{pBest}_i)) \\ & \text{or } (|f(\boldsymbol{pBest}_i) - f(\boldsymbol{p}_i)| < \eta f(\boldsymbol{pBest}_i) \\ & \text{and } \|\boldsymbol{wo}_{p_i}\| < \|\boldsymbol{wo}_{pBest_i}\|) \\ \boldsymbol{pBest}_i & else \end{cases} \quad (7)$$

$$\boldsymbol{gBest} = \begin{cases} \boldsymbol{p}_i & (f(\boldsymbol{gBest}) - f(\boldsymbol{p}_i) > \eta f(\boldsymbol{gBest})) \\ & \text{or } (|f(\boldsymbol{gBest}) - f(\boldsymbol{p}_i)| < \eta f(\boldsymbol{gBest}) \\ & \text{and } \|\boldsymbol{wo}_{p_i}\| < \|\boldsymbol{wo}_{gBest}\|) \\ \boldsymbol{gBest} & else \end{cases} \quad (8)$$

where $f(\boldsymbol{p}_i)$, $f(\boldsymbol{pBest}_i)$ and $f(\boldsymbol{gBest})$ are the fitness function value of fitness value of the position of the i-th particle, the best previous position of the i-th particle and the global best position of the swarm. $\boldsymbol{wo}_{p_i}$, $\boldsymbol{wo}_{pBest_i}$ and $\boldsymbol{wo}_{gBest}$ are the corresponding output weights of the position of the i-th particle, the best previous position of the i-th particle and the global best position obtained by MP inverse. In this way, particles with smaller fitness values or smaller norms are more likely to be selected as $\boldsymbol{pBest}_i$ or $\boldsymbol{gBest}$.

Step 4 calculates each particle's local attractor $\boldsymbol{P}_i$ and mean best position $\boldsymbol{mBest}$ according to equation (4) and (5).

Step 5 updates particle's new position according to equation (3)

Finally, we repeat Step 2 to Step 5 until the maximum optimization iterations are completed. Thus the network trained by ELM with the optimal input weights and hidden biases are obtained, and then the optimal network is applied to the benchmark problems.

In the proposed algorithm, each particle represents one possible solution to the optimization problem and is a combination of components with different meaning and different range.

All components of a particle are firstly initialized into the range [0, 1]. Therefore, before calculating, corresponding output weights and fitness evaluation in Step 2, they need to be converted to their real value.

For the input weights and bias, they are given by

$$z_{ij} = (z_l^{\max} - z_l^{\min}) p_{ij} + z_l^{\min} \quad (9)$$

Where $z_l^{\max} = 1$ and $z_l^{\min} = -1$ are the upper and lower bound for input weights and hidden bias are.

For s-parameters, they are given by

$$z_{ii} = round(p_{ij}) \quad (10)$$

where $round(\,)$ is a function that rounds to the nearest integer.

After the conversion of all variables, the fitness of each individual can be evaluated.

4 Performance evaluation

This section presents performance evaluations of QPSO-ELM on both benchmark regression problems and classification problems. The performance of QPSO-ELM is compared with ELM[2], GO-ELM[5], and IPSO-ELM[10]. Furthermore, in order to test how network structure optimization improves the generalization performance, IPSO-ELM incorporating hidden neurons optimization strategy (IPSO-ELM2) was tested on the same benchmark

problems.

All simulations have been made in Matlab R2008a environment running on a PC with 2.5 GHz CPU with 2 cores and 2 GB RAM. For the four optimized ELMs, the population size and the maximum number is very important. According to our tests, the number of the population is set as 100 and the maximum number of iterations is 50. The selection criteria for the two IPSO-ELMs and QPSO-ELM include the norm of output weights as Equation (7) and (8). The selection criteria for GO-ELM considers only the RMSE value or testing accuracy on validation data set and does not include the norm of output weights as suggested in Ref. [5]. Instead, GO-ELM incorporates Tikhonov's regularization in the least squares algorithm to improve the SLFN generalization capability.

In GO-ELM, the probability of crossover is 0.5 and the mutation probability is 10% as suggested in Ref. [5]. In the two PSO-ELMs, the inertial weight is set to decrease from 1.2 to 0.4 linearly with the iterations as suggested in Ref. [10]. In QPSO-ELM, the contraction-expansion coefficient β is set to decrease from 1.0 to 0.5 linearly with the iterations as suggested in Ref. [13]. The initial number of hidden neurons of all the optimized ELMs are set equal to that of ELM for each benchmark problem.

All the five ELMs are run 20 times separately for each benchmark problem and the results shown in the tables are the mean and standard deviation performance values in 20 trials. These performance values include training and testing RMSE, (training and testing accuracy for classification problems), mean number of hidden neurons, norm of output weights and condition number of hidden output matrix.

The condition number is a parameter qualitatively characterized the conditioning of a matrix. It is a good indicator to show how close a matrix is to be ill-conditioned. The smaller of the condition number, the better of the conditioning of matrix. It is given by

$$C(H)=\sqrt{\frac{\lambda_{\max}(H^TH)}{\lambda_{\min}(H^TH)}} \tag{11}$$

where $\lambda_{\min}(H^TH)$ and $\lambda_{\max}(H^TH)$ are the smallest and largest eigenvalues of the matrix H^TH.

4.1 Evaluation on regression problems

4.1.1 Function approximation

In this section, all the algorithms are compared on the approximation of the 'SinC' function:

$$y=\begin{cases}\sin(x)/x & x\neq 0\\ 1 & x=0\end{cases} \tag{12}$$

The training data set and testing data set have 5000 samples respectively and are available online[19]. Uniform noise randomly distributed in [–0.2, 0.2] has already been added to all the training samples while the testing data set remains noise-free. 20 hidden neurons are assigned for ELM as suggested in Ref. [2]. And the same is the initial number of hidden neurons for the four optimized ELMs

To avoid the over-fitting problem, the fitness of each particle is evaluated on the validation data set instead of the whole training data set. In this paper, 40% of the training data set was randomly selected and used as the validation data set and the left 60% was used as the 'actual' training data set. Tab. 1 shows the average performance of the five algorithms on 'SinC' function approximation problem. Fig. 1 shows the true and estimated output by QPSO-ELM on testing data set.

https://dc-china-simulation.researchcommons.org/journal/vol29/iss10/28
DOI: 10.16182/j.issn1004731x.joss.201710028

Tab. 1 Performance of the five algorithms on 'SinC' function approximation problem

Algorithms	Training time(s)	Training RMSE		Testing RMSE		Hidden neurons	Norm	Condition number
		Mean	Std	Mean	Std			
ELM	0.047	0.115 7	0.001 2	0.008 74	0.001 9	20	3.843 e+6	8.162 e+9
IPSO-ELM	84.52	0.115 2	0.006 1	0.008 35	0.002 5	20	7.252 e+4	3.186 e+8
IPSO-ELM2	94.56	0.115 0	0.007 5	0.008 01	0.004 6	14.6	**4.071 e+4**	2.854 e+8
GO-ELM	115.23	0.115 2	0.004 3	0.008 26	0.012 7	15.3	6.254 e+4	3.766 e+8
QPSO-ELM	96.3	**0.114 9**	0.001 1	**0.007 35**	0.004 2	**14.2**	4.428 e+4	**2.453 e+8**

It can be concluded from Tab. 1 that all the optimized ELMs obtain smaller mean RMSE values with less hidden neurons than ELM. At the same time, the norm of output weights and the condition number of the hidden matrix H obtained by the optimized ELMs are also smaller than those of ELM. This indicates that optimization algorithms would help ELM to attain a better generalization performance. IPSO-ELM2, GO-ELM and QPSO-ELM all obtain networks with fewer hidden neurons than ELM. This is mainly because the optimization of network structure helps them to achieve better performance with a much more compact network.

Also it can be observed clearly that the training time of ELM is much less than the optimized ones. Much of training time of the optimized ELMs is spent on evaluating all the individuals iteratively.

Among the four optimized ELMs, QPSO-ELM obtains the best training and testing RMSE and the condition number with the fewest hidden neurons. This suggests that QPSO-ELM is superior to the other optimized ELMs on 'SinC' function approximation problem. Fig. 2 shows that the estimated output is in good consistent with the desired output.

4.1.2 Real-world regression problems

The performances of all algorithms are also compared on two real-world regression problems from the UCI machine learning repository. They are Boston Housing data set and Abalone data set. Boston Housing data set concerns housing values in suburbs of Boston. It has 13 attributes and 506 instances. While Abalone data set concerns predicting the age of Abalone from 8 physical measurements and has 4177 instances. As there are no separate training data set and testing data set available for these two regression problems, for each trial of simulations, we randomly select half of the data as the testing data set and 20% of the data as the validation data set, the left data is used as training data set as suggested in Ref. [2].

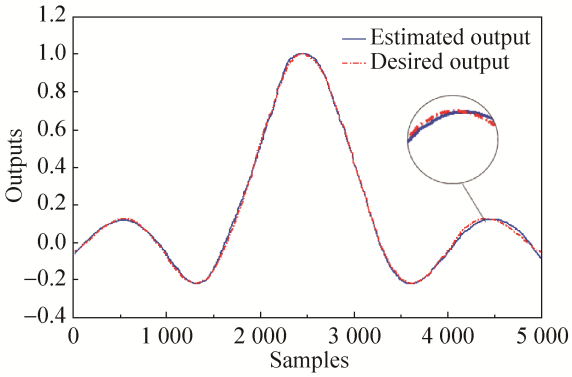


Fig. 2 Outputs of the QPSO-ELM learning algorithm on 'SinC' problem

The performances of the five algorithms on Boston Housing and Abalone problem are listed in Tab. 2-3.

As observed form Tab. 2-3, all the optimized ELMs need much more time than ELM which is largely because of the iteration nature of GA, PSO and QPSO. There is not much difference in training time among the four optimized ELMs and GO-ELM takes more time than the other three on both problems.

Tab. 2 Performance of the five algorithms on Boston Housing problem

Algorithms	Training time(s)	Training RMSE		Testing RMSE		Hidden neurons	Norm	Condition number
		Mean	Std	Mean	Std			
ELM	0.039	0.153 8	0.008 0	0.291 3	0.004 2	20	4.563 e+7	6.104 e+10
IPSO-ELM	49.23	0.142 9	0.010 2	0.228 1	0.019 5	20	3.977 e+6	5.014 e+8
IPSO-ELM2	44.56	0.124 3	0.011 7	0.219 4	0.005 7	18.2	2.913 e+6	2.355 e+8
GO-ELM	57.51	0.126 1	0.008 9	0.247 5	0.009 3	18.5	3.232 e+6	4.156 e+8
QPSO-ELM	55.98	**0.119 2**	0.009 1	**0.208 6**	0.005 6	**17.7**	**2.564 e+6**	**1.954 e+8**

Tab. 3 Performance of the five algorithms on Abalone problem

Algorithms	Training time(s)	Training RMSE		Testing RMSE		Hidden neurons	Norm	Condition number
		Mean	Std	Mean	Std			
ELM	0.0934	0.063 5	0.008 9	0.078 3	0.005 6	20	315.24	5 591.06
IPSO-ELM	179.53	0.032 6	0.012 8	0.053 5	0.004 2	20	151.87	2 056.31
IPSO-ELM2	185.60	0.031 8	0.009 7	0.042 9	0.004 4	18.1	122.39	**1 875.35**
GO-ELM	205.24	0.036 9	0.020 7	0.046 6	0.018 5	**17.6**	156.45	2 185.42
QPSO-ELM	198.91	**0.027 2**	0.008 4	**0.038 4**	0.005 2	17.9	**93.60**	1 956.36

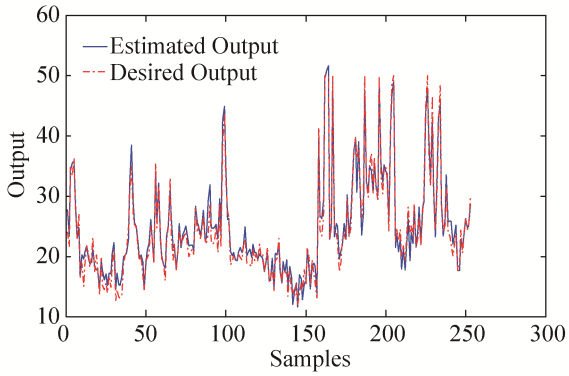


Fig. 3 Output of the QPSO-ELM algorithm on Housing problem

For testing RMSE, norm and condition number, the optimized ELMs obtain better results than ELM on the two data sets. Among the optimized ELMs QPSO-ELM performs the best on Boston Housing and obtains best Testing RMSE and norm value on abalone problem, IPSO-ELM2 obtain best condition number on abalone problem.

For the hidden neurons number, IPSO-ELM2, GO-ELM and QPSO-ELM achieve networks with fewer hidden neurons than ELM and IPSO-ELM, thus obtain a more compact network. GO-ELM performs the best on abalone data set and QPSO-ELM performs the best on Boston Housing data set.

In general, it can be concluded that QPSO-ELM is superior to the other optimized ELMs on these real-world regression problems. Fig. 3 shows the comparison of the desired and estimated output of the QPSO-ELM on Housing testing data set.

4.2 Evaluation on classification problems

The performances of all algorithms are tested on three real-world benchmark classification data sets which are listed in Tab. 4. The training data set, validation data set and testing data set are randomly generated at each trial of simulations according to the corresponding numbers in Tab. 4. The performances of the five algorithms on classification data sets are listed in Tab. 5-7.

It can be observed from these tables that number of hidden neurons assigned for ELM is different for different problems. For image segmentation and shuttle data set, 100 hidden neurons is enough for ELM to attain a good testing accuracy, while in

satellite image classification problem, if 100 hidden neurons are assigned for ELM, the average training accuracy is very poor (66.34%).

Therefore we assign 500 hidden neurons for ELM on satellite image data set as suggested in Ref. [2]. From Tab. 5, we can see that IPSO-ELM2, GO-ELM and QPSO-ELM greatly reduced the number of hidden neurons of the network and manage to maintain testing accuracy with the same level. QPSO-ELM attains the best testing accuracy with the fewest hidden neurons. In fact, QPSO-ELM with only 50 nodes can achieve similar generalization performance as ELM with 500 hidden neurons. And the number of hidden neurons is smaller than that of IPSO-ELM2 and GO-ELM. QPSO-ELM also performs the best in norm and condition number on satellite image classification.

Results on the other classification problems show the similar conclusions: all the optimized ELMs obtain better testing accuracy, norm and condition number. Except IPSO-ELM, all the optimized ELMs need fewer hidden neurons than ELM. Also all the optimized ELMs take much more training time than ELM.

QPSO-ELM performs the best in testing accuracy, hidden neurons, norm and condition number on shuttle problem. For Image segmentation problem, QPSO-ELM performs the best in testing accuracy, hidden neurons and norm. GO-ELM performs the best in condition number. In general, QPSO-ELM performs better than others on classification applications.

Tab. 4 Specification of four classification problems

Names	attributes	classes	Number of samples		
			Training set	Validation set	Testing set
satellite image	36	7	2 661	1 774	2 000
Image segmentation	19	7	1 000	524	786
Shuttle problem	9	7	26 100	17 400	14 500

Tab. 5 Performance of the five algorithms on satellite image classification

Algorithms	Training time(s)	Training accuracy		Testing accuracy		Hidden neurons	Norm	Condition number
		Mean	Std	Mean	Std			
ELM	0.033 5	0.873 1	0.010 6	0.852 3	0.008 3	500	352.43	5 062.41
IPSO-ELM	198.33	0.884 9	0.009 1	0.872 8	0.012 6	100	133.50	1 856.12
IPSO-ELM2	142.81	0.885 2	0.013 6	0.865 6	0.009 5	59.4	97.53	1 654.83
GO-ELM	274.07	0.877 3	0.009 2	0.869 0	0.017 4	62.0	126.58	1 545.76
QPSO-ELM	178.56	**0.890 5**	0.007 2	**0.876 5**	0.005 3	**49.7**	**48.61**	**895.49**

Tab. 6 Performance of the five algorithms on Image segmentation

Algorithms	Training time(s)	Training accuracy		Testing accuracy		Hidden neurons	Norm	Condition number
		Mean	Std	Mean	Std			
ELM	0.027	0.930 4	0.004 2	0.921	0.003 2	100	147.25	4 854.13
IPSO-ELM	39.08	0.938 0	0.006 3	0.947	0.012 4	100	96.34	2 029.72
IPSO-ELM2	38.34	0.954 2	0.006 8	0.958	0.009 6	70.2	85.43	**1 630.96**
GO-ELM	54.26	0.938 3	0.011 7	0.934	0.007 9	66.9	105.46	2 543.25
QPSO-ELM	45.36	**0.965 5**	0.004 3	**0.960**	0.008 2	**58.7**	**83.18**	1 891.38

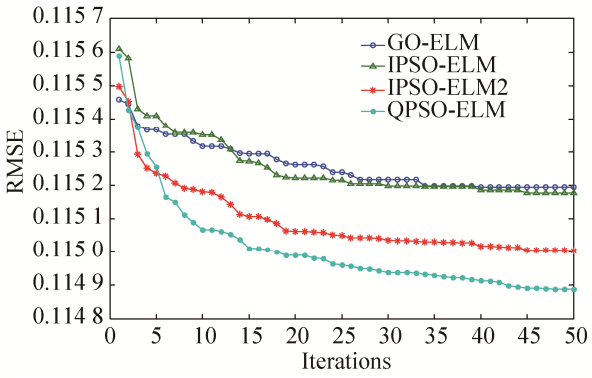
Tab. 7 Performance of the five algorithms on Shuttle problem

Algorithms	Training time(s)	Training accuracy		Testing accuracy		Hidden neurons	Norm	Condition number
		Mean	Std	Mean	Std			
ELM	7.293 7	0.956 9	0.019 6	0.958 3	0.018 2	100	15.580	1 640.94
IPSO-ELM	697.45	0.968 7	0.008 7	0.965 5	0.002 5	100	10.253	980.74
IPSO-ELM2	605.47	0.975 4	0.014 5	0.970 1	0.010 7	42.8	9.458	996.53
GO-ELM	816.50	0.977 1	0.016 9	0.964 8	0.005 6	31.6	11.025	1 114.25
QPSO-ELM	756.95	**0.985 7**	0.010 1	**0.979 3**	0.007 0	**29.8**	**8.964 7**	**954.030**

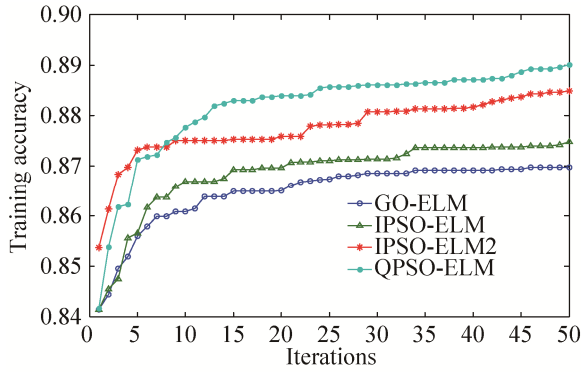
4.3 Further analysis

4.3.1 Comparison of convergence performance

In order to evaluate the proposed algorithm in depth, the mean evolution of the RMSE on validation dataset of 20 trials by the four optimized ELMs on 'SinC' function approximation and satellite image data sets are plotted in Fig. 4.



(a) 'SinC' function approximation



(b) Satellite image

Fig. 4. Mean evolution of the RMSE of the four algorithms

It can be observed from the Fig. 4(a) that QPSO-ELM has much better converge performance than the other three algorithms and obtains the best RMSE after 50 iterations, IPSO-ELM2 is better than GO-ELM and IPSO-ELM. In fact, QPSO-ELM can achieve the same RMSE level as IPSO-ELM2 with only half of the total iterations.

Fig. 4(b) shows the convergence of testing accuracy on satellite image problem. Similarly, our algorithm converges more effectively than the others and attains best results after iterations. It can be concluded that the introduction of quantum mechanics helps QPSO to search more effectively in search space, thus outperforms IPSO-ELM and GO-ELM in converging to a better RMSE result.

4.3.2 Comparison of norm and condition number

To show how our proposed algorithm improves the network condition and reduces the norm of the output weights, the norm and condition numbers of different algorithms on the two problems are shown as box plots in Fig. 5-6. Fig. 5 does not include results of ELM, as they are too much higher than that of optimized ELMs.

It can be observed from the box plots that all the optimized ELMs obtain smaller norm and condition numbers than ELM. Also the results obtained by optimized ELMs are more stable than ELM. This indicates that the selection criteria which includes both the fitness value (validation RMSE or testing accuracy) and the norm of output weights helps the algorithms obtain better generalization performance.

DOI: 10.16182/j.issn1004731x.joss.201710028

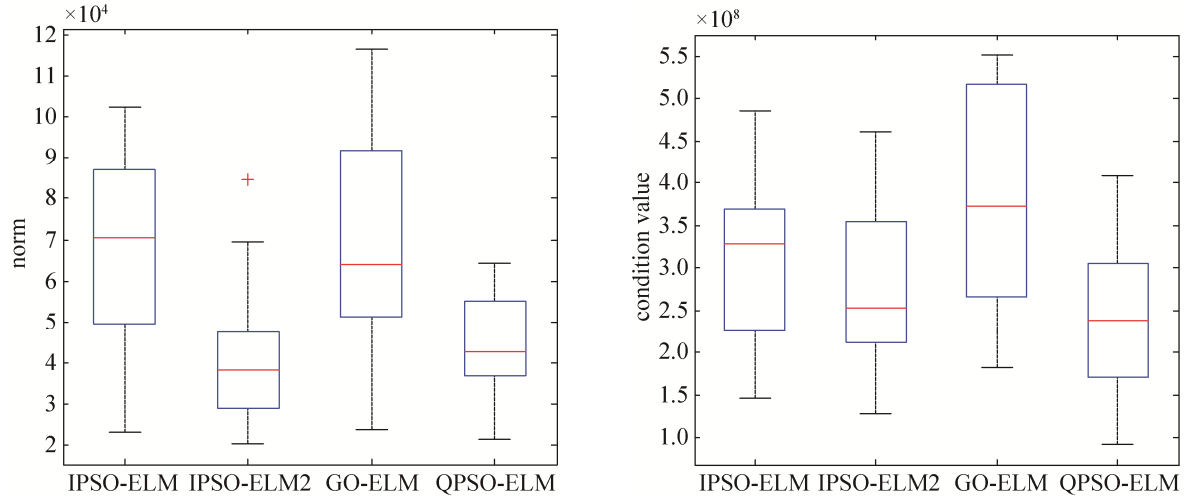


Fig. 5 Norm and condition number of SinC function approximation

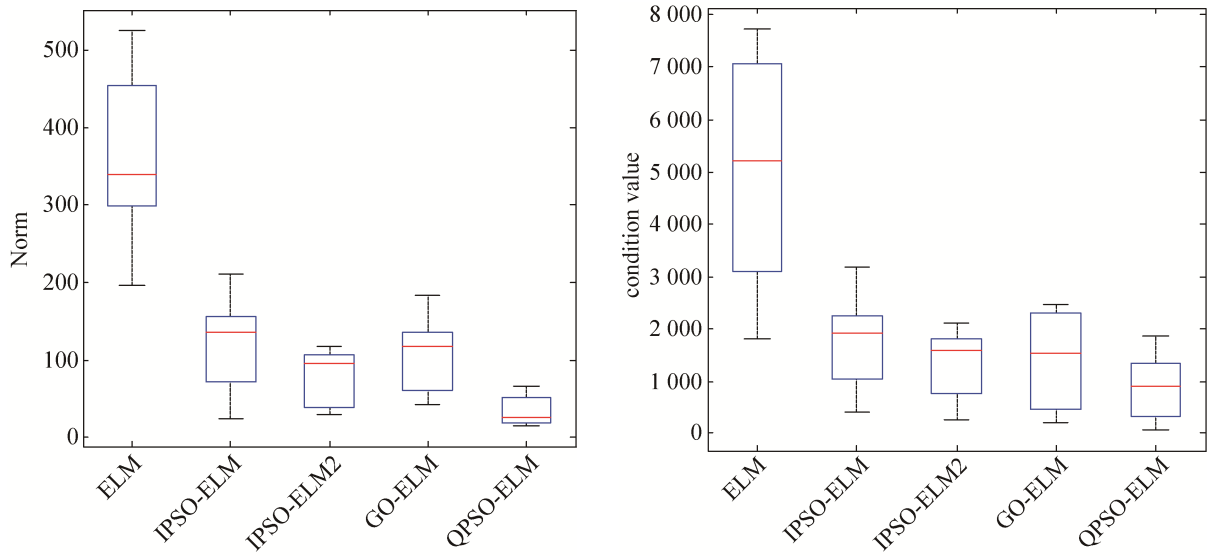


Fig. 6 Norm and condition number of satellite image problem

Among all the optimized algorithms, for the norm of output weights, QPSO-ELM performs the best on satellite image problem, IPSO-ELM2 attains best norm on 'SinC' function approximation. For condition number, QPSO-ELM performs the best on 'SinC' and satellite image problem. In general, QPSO-ELM performs better than the other optimized ELMs.

5 Conclusions

In this study, a new hybrid learning approach for SLFN named QPSO-ELM was proposed. The proposed approach optimizes both the neural network parameters (input weights and hidden biases) and hidden layer structure using QPSO. And the output weights are calculated by Moore-Penrose generalized inverse, like in the original ELM.

In the optimizing of network parameters, not only the RMSE on validation data set, but also the norm of the output weights are considered to be included in the selection criteria.

To validate the performance and effectiveness of the proposed approach, it was applied to some benchmark regression and classification problems.

Results show that the proposed approach has better generalization performance than the other optimized ELMs and maintains a well-conditioned system after training. Also, the proposed algorithm is more effective in reducing the network size, which in return further improves the generalization performance of the network.

References:

[1] G B Huang, Q Y Zhu, C K Siew. Extreme Learning Machine: a new learning scheme of feed forward neural networks [C]// 2004 International Joint Conference on Neural Networks, Budapest, Hungary. Piscataway, NJ, USA: IEEE, 2004: 985-990.

[2] G B Huang, Q Y Zhu, C K Siew. Extreme learning machine: theory and applications [J]. Neurocomputing (S0925-2312), 2006, 70(1/3): 489-501. doi:10.1016/j. neucom. 2005.12.126.

[3] Q Y Zhu, A K Qin, P N Suganthan, et al. Evolutionary extreme learning machine [J]. Pattern Recognition (S0031-3203), 2005, 38(10): 1759-1763. doi: 10.1016/j. patcog. 2005.03.028.

[4] Cao Z Lin, G B Huang. Self-adaptive evolutionary extreme learning machine [J]. Neural Processing Letter (S 1370-4621), 2012, 36(3): 285-305. doi:10.1007/s11063-012-9236-y.

[5] Tiago Matias, Francisco Souza, Rui Araújo, et al. Learning of a single-hidden layer feed forward neural network using an optimized extreme learning machine [J]. Neurocomputing (S0925-2312), 2014, 129: 428-436, doi: 10.1016/j.neucom.2013.09.016.

[6] C Chang-Yi, Y Yong-Chun. Solving of nonlinear equations based on PSO algorithm [J]. Computer Application and Software (S1000-386x), 2006, 23(5): 137-139.

[7] Z Miao, S Xie, Y Wu, et al. Aero-engine state variable modeling based on the improved particle swarm optimization [J]. Journal of Propulsion Technology (S 1001-4055), 2012, 33(1): 73-77.

[8] A H Gandomi, G J Yun, X-S Yang, et al. Chaos-enhanced accelerated particle swarm optimization [J]. Communications in Nonlinear Science and Numerical Simulation (S1007-5704), 2013, 18(2): 327-340, doi:10.1016/j. cnsns. 2012.07.017

[9] Y Xu, Y Shu. Evolutionary extreme learning machine based on particle swarm optimization [J]. Lecture Notes in Computer Science (S0302-9743), 2006, 36(3): 644-652.

[10] F Han, H Yao, Q Ling. An improved extreme learning machine based on particle swarm optimization [J]. Neurocomputing (S0925-2312), 2013, 116: 87-93, doi: 10.1016/j.neucom.2011.12.062.

[11] F vanden Bergh, A P Engelbrecht. A study of particle swarm optimization particle trajectories [J]. Information Sciences (S 0020-0255), 2006, 176(8): 937-971.

[12] J Sun, C H Lai, W B Xu, et al. A modified quantum-behaved particle swarm optimization [C]// Proceedings of the 7th International Conference on Computational Science (ICCS '07), Beijing, China. Germany: Springer, 2007: 294-301.

[13] Li R, Li W J, Zhang L, et al. An improved quantum-behaved particle swarm classifier based on weighted mean best position [C]// 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, Shanghai, China. USA: IEEE, 2009:327-331.

[14] Jianhua Xiao, Jin Xu, Zhihua Chen. A hybrid quantum chaotic swarm evolutionary algorithm for DNA encoding [J]. Computers and Mathematics with Applications (S0898-1221), 2009, 57(11-12): 1949-1958.

[15] D Chen, J Wang, F Zou, et al. An improved group search optimizer with operation of quantum-behaved swarm and its application [J]. Applied Soft Computing Journal (S 1568-4946), 2012, 2(2): 712-725.

[16] J Sun, W Fang, V Palade et al. Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point [J]. Applied Mathematics and Computation (S 0096-3003), 2011, 218(7): 3763-3775.

[17] R C Eberhart, J Kennedy. A new optimizer using particles swarm theory [C]// Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995. USA: IEEE, 1995: 39-43.

[18] R C Eberhart, J Kennedy. Particle swarm optimization. [C]// Proceeding of the IEEE International Conference on Neural Network, Perth, Australia, 1995. USA: IEEE, 1995: 1942-1948.

[19] G B Huang. Sinc.zip [DB/OL]. (2004-04-16) [2015-07-10].http://www.ntu.edu.sg/home/egbhuang/elm_codes.html.