Research Article

Real-Coded Quantum-Inspired Genetic Algorithm-Based BP Neural Network Algorithm

Jianyong Liu,¹ Huaixiao Wang,¹ Yangyang Sun,² Chengqun Fu,¹ and Jie Guo¹

¹College of Field Engineering, PLA University of Science and Technology, Nanjing 210007, China

²College of Defense Engineering, PLA University of Science and Technology, Nanjing 210007, China

Correspondence should be addressed to Huaixiao Wang; 753048237@qq.com

Received 19 November 2014; Revised 25 December 2014; Accepted 6 January 2015

Academic Editor: Sabri Arik

Copyright © 2015 Jianyong Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The method that the real-coded quantum-inspired genetic algorithm (RQGA) used to optimize the weights and threshold of BP neural network is proposed to overcome the defect that the gradient descent method makes the algorithm easily fall into local optimal value in the learning process. Quantum genetic algorithm (QGA) is with good directional global optimization ability, but the conventional QGA is based on binary coding; the speed of calculation is reduced by the coding and decoding processes. So, RQGA is introduced to explore the search space, and the improved varied learning rate is adopted to train the BP neural network. Simulation test shows that the proposed algorithm is effective to rapidly converge to the solution conformed to constraint conditions.

1. Introduction

Artificial neural networks (ANNs) are put forward to solve the nonlinear problem by simulating the operational process of nervous system. ANNs are powerful tools for prediction of nonlinearities [1] and with excellent nonlinear mapping ability, generalization, self-organization, and self-learning. ANNs have been widely applied in engineering and are steadily advancing into new areas [2].

The “feed-forward, back propagation” neural network (BPNN) is currently the most popular network architecture in use [3]. BPNN can be applied in a variety of fields according to the characteristics of the model. Sun et al. establish a prediction model based on improved BP neural network and adopt it to investigate quantitative evolution laws of equiaxed α in near- β forging of TA15 Ti-alloy [4]. Xiao et al. propose an approach of back propagation neural network with rough set for complicated short-term load forecasting with dynamic and nonlinear factors to develop the accuracy of predictions [5]. Wang et al. apply improved variable learning rate back propagation (IVL-BP) to short-term power load forecasting [6]. Yu et al. propose a dynamic all parameters adaptive BP neural networks model through fusion of genetic algorithm, simulated annealing algorithm, and the BP neural network and apply it to oil reservoir prediction [2].

Although system based on BPNN is with good performance, the BPNN lacks stability in some cases. The main faults of BPNN include the following. (1) Fixed learning rate leads to the slow convergence speed of network and long training time. (2) The gradient descent algorithm used to optimize the objective function in BPNN makes the computation overflow or swing between optimal values and does not make it converge to global optimal value [7]. (3) Structure and scale have much influence on the performance of BPNN. Different nodes in hidden layer and conversion functions acting on the same data will cause different results. (4) The convergence of BPNN is influenced by the choice of initial weights. Improper selection of the initial set of weights will make the algorithm trapped in local optimal solution. (5) The adjustment of weight and threshold values follows certain rule. It is impossible to adaptively adjust the structure in a fixed layout. (6) The learning algorithm of BPNN is based on back propagation of error, which leads to the slow learning convergence and easily falls into local minimum.

To overcome the abovementioned problems, many scholars have put forward some improved algorithm. Although the disadvantages are not ultimately overcome, the study level of BPNN is gradually increased. Different methods can be classified as follows.

(1) *The Optimization of Network Structure.* Experiential or statistical method is applied to determine the structure of BPNN, which is the optimal combination of the number of hidden layers, the number of hidden neurons, the choice of input factors, and parameters of the training algorithm [8]. The most systematic and general method is to utilize the principle of design experiment of Taguchi [9]. Use grey correlation analysis to determine the number of hidden nodes of the optimal network to improve the performance of the network [10]. Apply the network growth/removal algorithm to add/remove neurons from the initial structure according to the predetermined standard to represent the effect of changes on the performance of the ANNs. The basic rule is to increase neurons when the training process is slow or the mean square deviation is greater than the specified value. Decrease the number of neurons when the change of the number of neurons cannot change the response of the network accordingly or neuron weights remain unchanged for a long time. Growth/removal algorithm is the basic gradient descent method which cannot guarantee convergence to the global minimum; therefore, the algorithm may fall into local optimal solution near to the initial point. The structure can be changed by the application of genetic operators and evaluation of object function [11].

(2) *The Improved Training Method of BPNN.* Some new methods can be introduced to train the neural network; for example, the Online Gradient Method with changing scale can be used to train BPNN to achieve better convergence [12]. The real-coded chaotic quantum genetic algorithm is applied for training the fuzzy neural network to accelerate the convergence speed [13]. The transfer function, parameter, the accuracy of the assessment, and gradient descent coefficient can be improved properly in the process of training.

(3) *The Combination of BPNN with Other Optimization Algorithms to Optimize the Weights and Thresholds.* Zhuo et al. propose a simulated annealing- (SA-) genetic algorithm-BPNN-based color correction algorithm for traditional Chinese medicine tongue images [14]. Liu et al. apply GA-BP to predict bitterness intensity [15]. The GA-BP is compared with multiple linear regression, partial derivative least square regression, and BP method to prove the superiority of BP-GA model. Wang et al. improve BPNN by introducing cuckoo search algorithm to forecast lightning occurrence from sounding-derived indices over Nanjing [16].

Through the analysis, we can see that most optimization methods of the weights and thresholds of BPNN are based on the combination with GA or SA [14] or the combination of the three kinds of algorithms. There are some major disadvantages of GA and SA. When solving some complex optimization problems, the convergence speed of GA is slow, the convergence may be premature stagnation, and the numbers of species and individuals are large. SA utilizes the principle of crystallization process to minimum energy of metal to search for the minimum in general system. SA is first proposed in the literature of Kirkpatrick to be used to find the balanced combination of atoms set at a given temperature [17]. Compared with other methods, the main advantage of

FIGURE 1: Flowchart of training of BPNN.

the SA is the ability to avoid falling into local optimal value. SA is a random search algorithm, and better values and worse values can be obtained at a certain probability. However, the calculation amount of SA is large, especially for complex problems.

RQGA is a global optimization algorithm which can find the global optimal solution in the complex, multiple-extreme, and nondifferentiable vector space when the number of parameters is small. RQGA is with fast convergence speed and strong optimization ability and does not easily converge to the local optimal solution. Introducing RQGA to optimize the weights and thresholds of BPNN can guarantee getting better solution with higher probability.

2. BP Neural Network

BPNN is a kind of multilayer feed-forward network according to training of the error back propagation algorithm proposed by Rumelhart and McClland in 1986. The mean of “back propagation” is that the adjustment way of the weights of network is back propagation of error. As a result of the simple structure of BPNN, many adjustable parameters, and good maneuverability, the BPNN is one of the most widely used artificial neural network algorithms.

BPNN is a kind of typical forward network [18]. The training function is used reversely to change weights and thresholds through the positive transfer method of network structure. The samples to be measured are handled by the training model after training structure model of sample is established. The following formula is operating formula of BP where x is the input matrix, w is the weight matrix, and b is the threshold matrix:

$$f(x) = w^T \cdot x + b. \quad (1)$$

The specific process of BPNN is as shown in Figure 1.

(1) Initialization: determine weights w_{ij} , v_{jt} and thresholds θ_j , γ_t randomly.

(2) Select a group of input and output data $P_k = (a_1^k, a_2^k, \dots, a_n^k)$, $T_k = (s_1^k, s_2^k, \dots, s_p^k)$. Calculate the input of hidden layer s_j with the input sample $P_k = (a_1^k, a_2^k, \dots, a_n^k)$, weight value w_{ij} , and threshold θ_j . Then, calculate the output of hidden layer b_j with the translation function s_j , where $s_j = \sum_{i=1}^n w_{ij}a_i - \theta_j$, $j = 1, 2, \dots, p$, $b_j = f(s_j)$, $j = 1, 2, \dots, p$. Calculate the output of the output layer L_t with b_j , v_{jt} , and γ_t . Then calculate the response output of output layer with L_t by the translation function. Consider $L_t = \sum_{j=1}^p v_{jt}b_j - \gamma_t$, $t = 1, 2, \dots, q$, and $C_t = f(L_t)$, $t = 1, 2, \dots, q$.

(3) Calculate the total error of every output neuron d_t^k with the objective matrix of network $T_k = (y_1^k, y_2^k, \dots, y_q^k)$ and corresponding response output C_t . Consider $d_t^k = (y_t^k - C_t)C_t(1 - C_t)$, $t = 1, 2, \dots, q$. Then calculate the total error e_j^k with v_{jt} , d_t , and b_j . Consider $e_j^k = [\sum_{t=1}^q d_t v_{jt}] b_j (1 - b_j)$.

(4) If the error in (3) is smaller than the default error, end the training process; if not, correct v_{jt} and γ_t with d_t^k and b_j and correct w_{ij} and θ_j with e_j^k and $P_k = (a_1, a_2, \dots, a_n)$ at the same time.

(5) Select another group of input and output data randomly from the sample and return to step (2). Continue this process until the end condition is satisfied.

The learning rate of the conventional BP network is invariable. If the learning rate is too small, though the convergence can be guaranteed, the learning speed is slow; if the learning rate is large, it is easy to cause large fluctuation or deviation from the optimal solution. So the learning rate needs to be adjusted in the process of training.

The basic idea of variable learning rate is that if the average variance increases and exceeds the preset value after weights update, decrease the learning rate; if the average variance is less than the preset value, the learning rate stays unchanged; if the average variance is reduced, increase the learning rate.

3. Real-Coded Quantum-Inspired Genetic Algorithm

Conventional QGA is based on binary coding and can be used to solve the problem of combinatorial optimization well, such as the traveling salesman problem [19], knapsack problem [20, 21], and the filter design [22]. Using binary numbers to represent the parameters forces a trade-off between accuracy of representation and string lengths. RQGA is better in optimizing the real-valued problems with multiple-extremum. Optimization of weights and thresholds of BPNN is a typical optimization problem of real number. When the real number problem needs to be optimized, real encoding is considered better than binary and gray coding in solving the multiparameter optimization problem [23, 24]. RQGA has the inherent advantage of QGA. The search performance and quantum operators make RQGA have the characteristics of effectiveness, flexibility, robustness, and so on. The RQGA can solve the problem with real parameters well, so the RQGA is applied to optimize the BPNN.

3.1. Coding Method of RQGA. Initial chromosome includes string of real value and string of quantum bit value and is expressed as

$$\{P_1(t) \ P_2(t) \ \dots \ P_{Np}(t)\}, \text{ where,}$$

$$P_i(t) = \begin{Bmatrix} x_{i1}^t & x_{i2}^t & \dots & x_{iNg}^t \\ \theta_{i1}^t & \theta_{i2}^t & \dots & \theta_{iNg}^t \end{Bmatrix}, \quad i = 1, \dots, Np, \quad (2)$$

x_{ij}^t ($j = 1, \dots, Ng$) is the value of real number coding, and θ_{ij}^t ($j = 1, \dots, Ng$) denotes phase angle of quantum bit. So each chromosome contains information of real number space and phase space at the same time.

The characteristics of RQGA are as follows. (1) Quantum bit coding makes population obtain better diversity to reduce the calculation. (2) RQGA utilizes special quantum evolutionary operators to generate candidate solutions containing real parameters, which is different with the candidate solutions generated by quantum observation in QGA. (3) RQGA applies quantum rotation gate to realize the evolution of quantum bit, which is the same with QGA. (4) Migration of different quantum bits realizes the migration of population of different solution, so the convergence degree and the quality of the solution are improved.

The following method is utilized to generate real number candidate solution strings. There is a group of N_p quantum bit strings, Q_i^t ($i = 1, 2, \dots, N_p$), which is the i th quantum bit string in the t th generation. Correspondingly, there is the other group of N_p strings, P_i^t ($i = 1, 2, \dots, N_p$); each string contains N_g real number. There are N_g quantum bits in each Q_i representing the probability amplitude of α_i . The probability to generate the real number which is larger (smaller) than present number is determined by $|\alpha_i|^2$ ($|\beta_i|^2$). All the probabilities are equal at the beginning of search; α_i and β_i are initialized to 0.707. Every element of P_i ($i = 1, 2, \dots, N_p$) is initialized to a random number in the allowable range. Each pair of Q_i and P_i constitutes the i th family in the t th generation.

N_c solution strings p_{ij}^t ($j = 1, 2, \dots, N_c$) of the i th family are generated by Q_i^t , P_i^t , and P_{best}^t (the best solution found present). Fitness p_{ij}^t is calculated under the constraint condition. The process to generate P_{best}^t is shown in Figure 2.

3.2. Evolutionary Method of RQGA. Two kinds of neighbor operators are adopted to generate new strings. N_c neighbor solution strings are generated by Neighbor Operator 1 (NO1) and Neighbor Operator 2 (NO2) where the i th family is determined by the best neighbor solution C_i^t . If C_i^t is better than P_i^t , C_i^t replaces P_i^t to be P_i^{t+1} . If the optimal value of P_i^{t+1} is better than P_{best}^t , P_{best}^t is replaced by the optimal value of P_i^{t+1} to be P_{best}^{t+1} .

The basic principle of the two operators is as follows: NO1 has better search performance to generate solution strings which are very different from the given string. NO2 has better exploitation performance to make P_j converge to $P_{\text{best}j}$ in the process of algorithm. So, this algorithm can keep a balance between exploration and exploitation. The evolution

FIGURE 2: Diagrammatic sketch of generating new P_{best} .

of quantum bit represents the evolution of the superposition state, and change of $|\alpha|^2$ and $|\beta|^2$ is translated to the change of real number generated by the two neighbor operators. The role of the two neighbor operators is that NO1 is used to search the solution space and NO2 is used to converge to extreme value.

Neighbor Operator 1: in the t th generation, there are N_p quantum bit strings Q_i ; there are N_g elements in each quantum bit. NO1 generates solution string $P_{ij}^t (j = 1, 2, \dots, N_c)$; there are N_g elements in each string.

Generate an array R_{ij} with N_g elements; the value of elements of R_{ij} is +1 or -1 randomly. ρ_{ijk} is the k th element of R_{ij} ; then

$$\theta_{ijk}^t = \theta_{ijk}^{t-1} + \rho_{ijk} \delta, \quad (3)$$

where δ is the variety of angle, θ_{ijk}^t is the rotation angle, and $\theta_{ijk}^t = \arctan(\beta_{ijk}^t / \alpha_{ijk}^t)$. If $\rho_{ijk} = -1$, δ is a random number in the range $[0, \pi/2 + \theta_{ijk}^{t-1}]$; if $\rho_{ijk} = +1$, δ is a random number in the range $[0, \pi/2 - \theta_{ijk}^{t-1}]$.

The new formula of probability amplitude is $\begin{bmatrix} \alpha_{ijk}^t \\ \beta_{ijk}^t \end{bmatrix} = \begin{bmatrix} \cos \delta & \sin \delta \\ -\sin \delta & \cos \delta \end{bmatrix} \begin{bmatrix} \alpha_{ijk}^{t-1} \\ \beta_{ijk}^{t-1} \end{bmatrix}$.

The formula of individual elements is

$$P_{ijk}^t = (\alpha_{ijk}^t)^2 (P_{k \max} - P_{k \min}) + P_{k \min}, \quad (4)$$

FIGURE 3: Flowchart of using NO1 to calculate elements of individual.

where $P_{k \max}$ and $P_{k \min}$ are the maximum and minimum values in the allowable range. The flowchart of using NO1 to calculate the k th element of the j th individual of the i th population in the t th generation is shown in Figure 3.

TABLE 1: The method to determine the value of $\Delta\theta$ in the t th generation.

Fitness	Element value	$\Delta\theta$
X	$C_{ij} = P_{\text{BEST } j}$	0
$F(P) > F(P_{\text{BEST}})$	$C_{ij} > P_{\text{BEST } j}$	θ_t
	$C_{ij} < P_{\text{BEST } j}$	$-\theta_t$
$F(P) < F(P_{\text{BEST}})$	$C_{ij} > P_{\text{BEST } j}$	$-\theta_t$
	$C_{ij} < P_{\text{BEST } j}$	θ_t
$F(P) = F(P_{\text{BEST}})$	$C_{ij} > P_{\text{BEST } j}$	0
	$C_{ij} < P_{\text{BEST } j}$	0

θ_t is the size of the rotation angle in the t th generation, $\theta_t = \theta_{\max} \cdot \exp(-t/T)$, $\theta_{\max}$ is the maximal rotation angle, T is the total evolutionary generation, and t is the current evolutionary generation. This method to determine θ_t is helpful to the search in early stage and convergence precision in late stage.

Neighbor Operator 2: most of the mechanism of NO2 is the same with NO1. In addition, the value point generated by NO2 is between P_b and P_{best} , and the generated point is used for exploring the search space. Formula (4) is applied to calculate P_{ijk}^t in NO2 where $P_{k \max} = \max(P_{\text{bestik}}, C_i^t)$, $P_{k \min} = \min(P_{\text{bestik}}, C_i^t)$, and C_i^t is the best individual of the i th family in the t th generation.

NO1 is better in exploration; NO2 is better in exploitation. Exploration is important in the early evolution; exploitation is important in the late evolution. So NO1 is carried out with a greater frequency in the early evolution, and NO2 is carried out with a greater frequency in the late evolution. The specific frequency is defined by formula (9). Consider

$$F_{\text{NO1}} = \frac{(90 - 80t/T)}{100}, \quad (5)$$

$$F_{\text{NO2}} = \frac{(10 + 80t/T)}{100},$$

where F_{NO1} is the use frequency of NO1, F_{NO2} is the use frequency of NO2, t is the current evolutionary generation, and T is the total evolutionary generation of every circle.

3.3. Update of Quantum Bit String. The individual states of all quantum bits in Q are changing during the process of update so that the probability of the generated solution similar to the current optimal solution increases gradually. The change of probability is determined by the learning rate $\Delta\theta$. The value of $\Delta\theta$ under different conditions is shown in Table 1.

$\Delta\theta$ determines the speed of the quantum bit changing from 0.707 to the final value of 0 or 1. $\Delta\theta$ needs to be small enough to ensure that the number of generation that α changes from 0.707 to 0 (or 1) is large enough. So, the probability to generate the solution similar to the current best solution is larger when most of the quantum bits converge to 0 or 1.

Two kinds of migration (global and local) take effect together. P^t is selected randomly to update some Q_i^t in local migration; P_{best} is used to update Q_i^t in the global migration. It also needs to consider the particularity of the

FIGURE 4: Flowchart of RQGA.

application object when RQGA is applied to specific problem. The flowchart of RQGA is shown in Figure 4.

RQGA is with good global search ability. RQGA is usually not restricted by the constraint conditions such as the property of the problem and the structure of the model of the problem and can converge to global optimal solution with a larger probability. Robustness of RQGA makes it capable of being combined with BP algorithm to improve the generalization ability and learning ability of neural network. And the encoding manner based on real number avoids encoding and decoding in binary encoding manner to improve the computational efficiency.

4. The BPNN Based on RQGA

The convergence speed of BPNN is slow, so the RQGA is introduced to optimize the parameters of network, speed up the convergence, and obtain the global optimal solution.

TABLE 2: The training sample data.

Sample order	Characteristic value of sample	State
1	0.2286, 0.1292, 0.072, 0.1592, 0.1335, 0.0733, 0.1159, 0.094, 0.0522, 0.1345, 0.009, 0.126, 0.3619, 0.069, 0.1828	Normal
2	0.209, 0.0947, 0.1393, 0.1387, 0.2558, 0.09, 0.0771, 0.0882, 0.0393, 0.143, 0.0126, 0.167, 0.245, 0.0508, 0.1328	Normal
3	0.0442, 0.088, 0.1147, 0.0563, 0.3347, 0.115, 0.1453, 0.0429, 0.1818, 0.0378, 0.0092, 0.2251, 0.1516, 0.0858, 0.067	Normal
4	0.2603, 0.1715, 0.0702, 0.2711, 0.1491, 0.133, 0.0968, 0.1911, 0.2545, 0.0871, 0.006, 0.1793, 0.1002, 0.0789, 0.0909	Crack
5	0.369, 0.2222, 0.0562, 0.5157, 0.1872, 0.1614, 0.1425, 0.1506, 0.131, 0.05, 0.0078, 0.0348, 0.0451, 0.0707, 0.088	Crack
6	0.0359, 0.1149, 0.123, 0.546, 0.1977, 0.1248, 0.0624, 0.0832, 0.164, 0.1002, 0.0059, 0.1503, 0.1837, 0.1295, 0.07	Crack
7	0.1759, 0.2347, 0.1829, 0.1811, 0.2922, 0.0655, 0.0774, 0.02273, 0.2056, 0.0925, 0.0078, 0.1852, 0.3501, 0.168, 0.2668	Defect
8	0.0724, 0.1909, 0.134, 0.2409, 0.2842, 0.045, 0.0824, 0.1064, 0.1909, 0.1586, 0.0116, 0.1698, 0.3644, 0.2718, 0.2494	Defect
9	0.2634, 0.2258, 0.1165, 0.1154, 0.1074, 0.0657, 0.061, 0.2623, 0.2588, 0.1155, 0.005, 0.0978, 0.1511, 0.2273, 0.322	Defect

TABLE 3: The test sample data.

Sample order	Characteristic value of sample	State
1	0.2101, 0.095, 0.1298, 0.1359, 0.2601, 0.1001, 0.0753, 0.089, 0.0389, 0.1451, 0.0128, 0.159, 0.2452, 0.0512, 0.1319	Normal
2	0.2593, 0.18, 0.0711, 0.2801, 0.1501, 0.1298, 0.1001, 0.1891, 0.2531, 0.0875, 0.0058, 0.1803, 0.0992, 0.0802, 0.1002	Crack
3	0.2599, 0.2235, 0.1201, 0.0071, 0.1102, 0.0683, 0.0621, 0.2597, 0.2602, 0.1167, 0.0048, 0.1002, 0.1521, 0.2281, 0.3205	Defect

RQGA is a global search process, which searches from one population to another. The parameter space is sampled unceasingly and the direction of the search is toward the area of the current optimal solution. The BPNN based on RQGA (RQGA-BP) combines advantages of RQGA and BPNN. RQGA is applied to optimize the weights and thresholds of input layer and hidden layer to avoid BP algorithm trapped in local minima.

The process of RQGA optimizing BPNN is divided into three parts: structure determination of BPNN, RQGA optimization, and BPNN optimization. Structure determination of BPNN needs to specify the number of parameters of input and output. RQGA optimization optimizes the weights and thresholds of BPNN. Each individual of the population contains all the weights and thresholds of network. Flowchart of RQGA-BP is shown in Figure 5.

The specific steps of RQGA-BP are as follows. (1) Code parameters. Set the weights and thresholds as genes. Each weight and threshold is expressed by a real number. So the evolution is based on some weights and thresholds. (2) Generate initial population. The range of gene is $(-0.5, 0.5)$ because the weights are small in good network. (3) Calculate fitness. The goal of BPNN is to make the residual error between forecast value and expected value as small as possible, so the norm of error matrix between expected value and forecast value is set to be the output of the objective function. Adopt sort of fitness assignment function with the manner of linear ranking and differential pressure 2 to estimate the fitness. (4) Evolve the population with RQGA and calculate the fitness of new individual. (5) The gene of best individual is the optimal solution of the weights and thresholds and is used to predict neural network.

5. Case Analysis

Two cases are used to justify the performance of RQGA-BP.

FIGURE 5: Flowchart of BPNN optimized by RQGA.

5.1. Case One. Apply RQGA-BP to fault detection for parts of machinery. Due to invisibility of the fault of mechanical parts, the possible states can be concluded by measuring the related characteristic parameters. Select 15 characteristic parameters of parts and normalize the data. The training sample data and the test sample data are shown in Tables 2 and 3, respectively.

The formula $n_2 = 2 \times n_1 + 1$ is adopted to calculate the number of neurons in hidden layer. The transfer function of

TABLE 4: The experiment results of the three algorithms.

Algorithm	Error	1	2	3	4	5	6	7	8	9	10	Average value
BP	Error 1	0.9445	1.0242	1.1473	1.0338	0.8463	0.9731	1.0185	1.0937	0.9765	1.0355	1.0093
	Error 2	1.9459	1.9504	2.0456	1.9521	1.8376	1.9582	1.9483	2.0647	1.9539	1.9568	1.9614
GA-BP	Error 1	0.8928	0.3193	0.53883	0.6493	0.9478	0.7485	0.3847	0.4838	0.5235	0.8384	0.6327
	Error 2	1.6035	1.321	1.3559	1.4857	1.9434	1.6683	1.3641	1.4759	1.4937	1.7758	1.5487
RQGA-BP	Error 1	0.5463	0.3027	0.4394	0.3859	0.5673	0.4375	0.5375	0.6473	0.3532	0.3391	0.4556
	Error 2	1.5274	1.3267	1.4589	1.3845	1.4838	1.3423	1.5384	1.4873	1.3839	1.3868	1.432

the neurons in hidden layer is S-shaped tangent function. The transfer function of the neurons in output layer is S-shaped logarithmic function. The states of parts are divided into three kinds of situation and the output form of the three situations is as follows:

Normal: (1, 0, 0); Crack: (0, 1, 0); Defect: (0, 0, 1).

The nodes number of input layer is 15, nodes number of output layer is 3, and nodes number of hidden layer is 31. So the number of weights is 558, the number of thresholds is 34, and the number of total parameters that needs to be optimized is 592. The training frequency of network is 1000, the training goal is 0.01, and the learning rate is 0.1. The norm of test error of test sample is set as the quota to measure the generalization capability of network. The fitness value of individual is calculated based on the norm of error. The smaller the error is, the larger the fitness of individual is.

The initial code of RQGA-BP is as follows:

$$\{P_1(t) \ P_2(t) \ \cdots \ P_i(t) \ \cdots \ P_{40}(t)\}, \text{ where} \quad (6)$$

$$P_i(t) = \begin{Bmatrix} x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{i592} \\ \theta_{i1} & \theta_{i2} & \cdots & \theta_{ij} & \cdots & \theta_{i592} \end{Bmatrix},$$

x_{ij} is a random number in the range $[-0.5, 0.5]$ and $\theta_{ij} = \pi/4$.

Apply BP, GA-BP, and RQGA-BP to process the data. The size of population is 40, and the maximal evolution generation is 50. Table 4 shows the results of the experiment by performing each algorithm 10 times, where Error 1 stands for the training error and Error 2 stands for the test error. We can get the conclusion that the test result of training and prediction data of RQGA-BP is better than GA-BP and the test result of training and prediction data of GA-BP is better than BP.

Apply the test method of mean difference of two normal populations (t -test) to analyze the error of BP, GA-BP, and RQGA-BP.

The data of error of the three methods can be considered to be samples from normal population $N(\mu, \sigma^2)$. The means of the sample of the three methods are $\bar{X}_1$, $\bar{X}_2$, and $\bar{X}_3$, respectively, and the variances are S_1^2 , S_2^2 , and S_3^2 . The t -statistic is introduced to be the test statistic:

$$t = \frac{\bar{X}_i - \bar{X}_j - \delta}{S_w \sqrt{1/n_i + 1/n_j}}, \quad (7)$$

where $\delta = \mu_i - \mu_j$, $S_w^2 = ((n_i - 1)S_i^2 + (n_j - 1)S_j^2)/(n_i + n_j - 2)$, and $S_w = \sqrt{S_w^2}$.

The form of rejection region is $|((\bar{x}_i - \bar{x}_j) - \delta)|/s_w \sqrt{1/n_i + 1/n_j} \geq k$.

$\{\text{Reject } H_0 \text{ when } H_0 \text{ is true}\} = P_{\mu_i - \mu_j = \delta}\{|((\bar{x}_i - \bar{x}_j) - \delta)|/s_w \sqrt{1/n_i + 1/n_j} \geq k\} = \alpha$; then $k = t_{\alpha/2}(n_i + n_j - 2)$.

So the rejection region is $|t| = |((\bar{x}_i - \bar{x}_j) - \delta)|/s_w \sqrt{1/n_i + 1/n_j} \geq t_{\alpha/2}(n_i + n_j - 2)$.

The hypotheses $H_0: \mu_i - \mu_j < 0$ and $H_1: \mu_i - \mu_j \geq 0$ need to be tested.

The mean and variance of Error 1 of BP, GA-BP, and RQGA-BP are as follows: $n_1 = 10$, $\bar{x}_1 = 1.0093$, $s_1^2 = 0.0068$; $n_2 = 10$, $\bar{x}_2 = 0.6327$, $s_2^2 = 0.0473$; $\bar{x}_3 = 0.4556$, $s_3^2 = 0.0130$. Then, $s_{w1}^2 = 0.02705$, $s_{w2}^2 = 0.0302$.

Consider $t_1 = 5.1201 > t_{0.005}(18) = 2.8784$. So reject H_0 ; that is, Error 1 of BP is larger than the GA-BP with probability of more than 99.5%.

Consider $t_2 = 2.2788 > t_{0.025}(18) = 2.1009$. So reject H_0 ; that is, Error 1 of GA-BP is larger than the RQGA-BP with probability of more than 97.5%.

The mean and variance of Error 2 of BP, GA-BP, and RQGA-BP are as follows: $n_1 = 10$, $\bar{x}_1 = 1.9614$, $s_1^2 = 0.0038$; $n_2 = 10$, $\bar{x}_2 = 1.5487$, $s_2^2 = 0.0400$; $\bar{x}_3 = 1.432$, $s_3^2 = 0.0058$. Then, $s_{w1}^2 = 0.0219$, $s_{w2}^2 = 0.0239$.

Consider $t_1 = 6.2358 > t_{0.005}(18) = 2.8784$. So reject H_0 ; that is, Error 2 of BP is larger than the GA-BP with probability of more than 99.5%.

Consider $t_2 = 1.6879 > t_{0.10}(18) = 1.3304$. So reject H_0 ; that is, Error 2 of GA-BP is larger than the RQGA-BP with probability of more than 90%.

Select one evolution of GA-BP and RQGA-BP randomly; the change process of error is shown in Figure 6. The x -axis represents the evolutionary generation; the y -axis represents the error. The active line represents the evolutionary process of GA-BP. The dotted line represents the evolutionary process of RQGA-BP. It can be concluded from the figure that the final error of RQGA-BP is smaller, and the evolution of RQGA-BP is faster.

5.2. Case Two. Apply RQGA-BP to forecast the gasoline octane number. There are some disadvantages in traditional experimental method to measure the octane number, for example, the large dosage of sample, long experimental period, and high cost. Near infrared spectroscopy (NIR) is with the advantages of lower cost, no pollution, nondestructive testing, and online analysis. Use Fourier transform near infrared transform spectrometer to scan 60 groups of gasoline

TABLE 5: Result of GA-BPNN and RQGA-BPNN.

Number of times	Parameters			
	E_z (GA-BP)	E_z (RQGA-BP)	R^2 (GA-BP)	R^2 (RQGA-BP)
1	0.0344	0.0326	0.9603	0.9782
2	0.0364	0.0245	0.9425	0.9760
3	0.0326	0.0275	0.9311	0.9728
4	0.0225	0.0295	0.9788	0.9859
5	0.0538	0.0284	0.9031	0.9785
6	0.0447	0.0299	0.9188	0.9745
7	0.0216	0.0283	0.9704	0.9518
8	0.0325	0.0323	0.8515	0.9488
9	0.0529	0.0298	0.8912	0.9496
10	0.0319	0.0218	0.9325	0.9680

FIGURE 6: The evolutionary process of GA-BP and RQGA-BP.

sample. The scanning range is 900–1700 nm, scanning interval is 2 nm, and each sample contains 401 wavelength points. Diagram of NIR of samples is as shown in Figure 7.

Utilize the laboratory testing method to test the octane number of sample. Apply GA-BPNN and RQGA-BPNN to set up the mathematical model between the near infrared spectrum and the octane number, respectively. Number of neurons of input layer is 401; number of neurons of output layer is 1. Since the number of neurons of input layer is large, adopt formula $n_2 = \log_2^{n_1}$ to calculate the number of neurons of hidden layer to 9. The number of training times is 1000, training target is 0.001, and learning rate is 0.01.

Apply randomly select method to generate training set and test set from 60 samples. The number of training sets is 50, and the number of test sets is 10. Since the training set and testing set are randomly generated each time, the results of each run may be different from each other.

After the test, evaluate the generalization ability of the network by calculating deviation between predicted value

FIGURE 7: Diagram of NIR of samples.

and true value. Choose relative error and the coefficient of decision to evaluate the generalization ability. The calculation formulas are shown in formula (8) and formula (9), respectively:

$$E_i = \frac{|\hat{y}_i - y_i|}{y_i}, \quad i = 1, 2, \dots, n, \quad (8)$$

$$R^2 = \frac{l \sum_{i=1}^l \hat{y}_i y_i - \sum_{i=1}^l \hat{y}_i \sum_{i=1}^l y_i^2}{l \sum_{i=1}^l \hat{y}_i^2 - \sum_{i=1}^l \hat{y}_i^2 l \sum_{i=1}^l y_i^2 - \sum_{i=1}^l y_i^2}, \quad (9)$$

$$E_z = \sum_{i=1}^n E_i, \quad (10)$$

where $\hat{y}_i$ ($i = 1, 2, \dots, n$) is the prediction value of the i th sample; y_i ($i = 1, 2, \dots, n$) is the true value of the i th sample; n is the number of samples. The smaller the sum E_z of relative error E_i the better, and the larger the decision coefficient $R^2 \in [0, 1]$ the better. Run GA-BPNN and RQGA-BPNN 10 times, respectively; the result is shown in Table 5.

Apply the t -test method in Case One to compare the performance of E_z and R^2 of GA-BP and RQGA-BP.

TABLE 6: Result of the fourth experiment.

Order number	Parameters				
	Value of laboratory testing method	Prediction value of GA-BP	Prediction value of RQGA-BP	Error value of GA-BP	Error value of RQGA-BP
1	88.6500	88.1141	88.8022	0.0060	0.0017
2	88.8500	88.9801	89.1007	0.0015	0.0028
3	87.6000	87.4219	87.7384	0.0020	0.0016
4	85.2500	85.1432	85.1895	0.0013	0.0007
5	85.1000	85.4010	85.3730	0.0035	0.0032
6	88.1000	88.2361	88.6818	0.0015	0.0066
7	88.5500	88.4808	88.5914	0.0008	0.0005
8	88.9000	88.7857	89.2359	0.0013	0.0038
9	88.2000	88.0418	88.7898	0.0018	0.0067
10	85.3000	85.0685	85.4639	0.0027	0.0019

FIGURE 8: Result of the fourth experiment.

The mean and variance of E_z of GA-BP and RQGA-BP are as follows: $n_1 = 10$, $\bar{x}_1 = 0.0363$, $s_1^2 = 0.00012$; $n_2 = 10$, $\bar{x}_2 = 0.0285$, $s_2^2 = 0.0000109$. Then, $s_w^2 = 0.00006545$.

Consider $t = 2.1559 > t_{0.025}(18) = 2.1009$. So reject H_0 ; that is, E_z of GA-BP is larger than the RQGA-BP with probability of more than 97.5%.

The mean and variance of R^2 of GA-BP and RQGA-BP are as follows: $n_1 = 10$, $\bar{x}_1 = 0.9280$, $s_1^2 = 0.0015$; $n_2 = 10$, $\bar{x}_2 = 0.9684$, $s_2^2 = 0.0002$. Then, $s_w^2 = 0.00085$.

Consider $t = -3.098 \leq -t_{0.005}(18) = -2.8784$. So accept H_0 ; that is, R^2 of GA-BP is smaller than the RQGA-BP with probability of more than 99.5%.

As can be seen from the data in the table, the seventh result of GA-BP is better than RQGA-BP; the remaining results of RQGA-BP are better than GA-BP.

Set the fourth result as the example; the specific calculation results are as shown in Table 6.

The contrast of prediction results of two methods with true value of the fourth experiment is as shown in Figure 8.

It can be seen that both GA-BP and RQGA-BP can predict the octane content and the result of RQGA-BP is better.

6. Conclusions

The optimization of weights and thresholds of BPNN is a numerical optimization problem. The purpose of RQGA optimizing BPNN is to obtain better initial weights and thresholds through RQGA. The individual in RQGA represents the initial weights and thresholds of network, and the norm of test error of the prediction sample is the output of objective function. Compared with conventional BPNN, the RQGA-BP is with higher convergence rate.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] B. H. M. Sadeghi, "A BP-neural network predictor model for plastic injection molding process," *Journal of Materials Processing Technology*, vol. 103, no. 3, pp. 411–416, 2000.
- [2] S. Yu, K. Zhu, and F. Diao, "A dynamic all parameters adaptive BP neural networks model and its application on oil reservoir prediction," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 66–75, 2008.
- [3] Y. L. Loukas, "Artificial neural networks in liquid chromatography: efficient and improved quantitative structure-retention relationship models," *Journal of Chromatography A*, vol. 904, no. 2, pp. 119–129, 2000.
- [4] Z. Sun, X. Wang, J. Zhang, and H. Yang, "Prediction and control of equiaxed α in near- β forging of TA15 Ti-alloy based on BP neural network: for purpose of tri-modal microstructure," *Materials Science and Engineering A*, vol. 591, pp. 18–25, 2014.
- [5] Z. Xiao, S.-J. Ye, B. Zhong, and C.-X. Sun, "BP neural network with rough set for short term load forecasting," *Expert Systems with Applications*, vol. 36, no. 1, pp. 273–279, 2009.

- [6] Y. Wang, D. Niu, and L. Ji, "Short-term power load forecasting based on IVL-BP neural network technology," *Systems Engineering-Integrating Procedia*, vol. 4, pp. 168–174, 2012.
- [7] L. Zhang and G. Subbarayan, "An evaluation of back-propagation neural networks for the optimal design of structural systems: part I. Training procedures," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 25–26, pp. 2873–2886, 2002.
- [8] P. G. Benardos and G.-C. Vosniakos, "Prediction of surface roughness in CNC face milling using neural networks and Taguchi's design of experiments," *Robotics and Computer-Integrated Manufacturing*, vol. 18, no. 5–6, pp. 343–354, 2002.
- [9] J. P. Ross, *Taguchi Techniques for Quality Engineering*, McGraw-Hill, New York, NY, USA, 1996.
- [10] W.-M. Tang, "The study of the optimal structure of BP neural network," *System Engineering—Theory and Practice*, vol. 25, no. 10, pp. 95–100, 2005.
- [11] H. Niska, T. Hiltunen, A. Karppinen, J. Ruuskanen, and M. Kolehmainen, "Evolving the neural network model for forecasting air pollution time series," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 2, pp. 159–167, 2004.
- [12] W. Wu, G. Feng, Z. Li, and Y. Xu, "Deterministic convergence of an online gradient method for BP neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 533–540, 2005.
- [13] S. Zhao, G. Xu, T. Tao, and L. Liang, "Real-coded chaotic quantum-inspired genetic algorithm for training of fuzzy neural networks," *Computers and Mathematics with Applications*, vol. 57, no. 11–12, pp. 2009–2015, 2009.
- [14] L. Zhuo, J. Zhang, P. Dong, Y. Zhao, and B. Peng, "An SA-GA-BP neural network-based color correction algorithm for TCM tongue images," *Neurocomputing*, vol. 134, pp. 111–116, 2014.
- [15] R. Liu, X. Zhang, L. Zhang et al., "Bitterness intensity prediction of berberine hydrochloride using an electronic tongue and a GA-BP neural network," *Experimental and Therapeutic Medicine*, vol. 7, no. 6, pp. 1696–1702, 2014.
- [16] J. Wang, Z. Sheng, B. Zhou, and S. Zhou, "Lightning potential forecast over Nanjing with denoised sounding-derived indices based on SSA and CS-BP neural network," *Atmospheric Research*, vol. 137, pp. 245–256, 2014.
- [17] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [18] L. Ke, G. Wenyan, S. Xiaoliu, and T. Zhongfu, "Research on the forecast model of electricity power industry loan based on GA-BP neural network," *Energy Procedia*, vol. 14, pp. 1918–1924, 2012.
- [19] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 61–66, 1996.
- [20] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, vol. 2, pp. 1354–1360, La Jolla, Calif, USA, July 2000.
- [21] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.
- [22] S. Zhou, W. Pan, B. Luo, W. L. Zhang, and Y. Ding, "A novel quantum genetic algorithm and its application," *Acta Electronica Sinica*, vol. 32, pp. 476–479, 2004.
- [23] H. Chen, J. S. Zhang, and C. Zhang, "Chaos updating rotated gates quantum-inspired genetic algorithm," in *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS '04)*, vol. 2, pp. 1108–1112, June 2004.
- [24] G. S. Sailesh Babu, D. Bhagwan Das, and C. Patvardhan, "Real-parameter quantum evolutionary algorithm for economic load dispatch," *IET Generation, Transmission and Distribution*, vol. 2, no. 1, pp. 22–31, 2008.