Training of Fuzzy Neural Networks via Quantum-Behaved Particle Swarm Optimization and Rival Penalized Competitive Learning

Saeed Farzi
Department of Computer Engineering, Islamic Azad University, Iran

Abstract: *There are some difficulties encountered in the application of fuzzy Radial Basis Function (RBF) neural network. One of them is how to determine the number of hidden rule neurons and another difficulty is about interpretability. In order to overcome these difficulties, we have proposed a fuzzy neural network based on RBF network and takagi-sugeno fuzzy system. We have used a new structure of fuzzy RBF neural network, which has been proved that it is better than other structures in term of interpretability. Our model also use a Rival Penalized Competitive Learning (RPCL) and a swarm based algorithm called Quantum-behaved Particle Swarm Optimization (QPSO) to determine design parameters of hidden layer and design parameters of output layer, respectively. RPCL is the best clustering algorithm that is introduced so far. The Particle Swarm Optimization (PSO) is a well-known population-based swarm intelligence algorithm. The QPSO is also proposed by combining the classical CPSO philosophy and quantum mechanics to improve performance of PSO. We have compared the performance of the proposed method with gradient based method. Simulation results of nonlinear function approximation demonstrate the superiority of the proposed method over gradient based method.*

Keywords: *Fuzzy RBF, RPCL, PSO.*

Received January 22, 2010; accepted August 10, 2010

1. Introduction

Fuzzy neural network is an intelligent neuro-fuzzy technique used to model and control of ill-defined and uncertain systems. It is based on the input-output data pairs of the system under consideration. Fuzzy inference and neural network have been combined to integrate the excellent learning capability of neural networks with fuzzy inference systems, resulting in neuro-fuzzy modeling approaches that combine the benefits of these two powerful paradigms into a single system and provide a powerful framework to extract good-quality fuzzy rules from numerical data [7, 16].

Jang and Sun [3] have shown that Radial Basis Function (RBF) networks and a simplified class of fuzzy systems are functionally equivalent under some mild conditions. This functional equivalence has made it possible to combine the features of these two systems, which has been developed into a powerful type of neuro-fuzzy systems [5].

Unfortunately, there are some difficulties encountered in the application of fuzzy RBF network. One of them is how to determine the number of hidden rule neurons, which is also called network structure learning problem [1, 8]. On one hand, fuzzy RBF network is unable to fulfil a given task if there are too few hidden neurons. On the other hand, too many hidden neurons not only increase the computation time, but also weaken the network generalization performance. And another difficulty is about interpretability. Traditionally, fuzzy neural networks are trained by using gradient-based methods, which may fall into a local minimum during the process and lose its interpretability or transparency, which is one of the most important features of fuzzy systems.

In order to overcome these difficulties, we propose a fuzzy neural network based on RBF network and Takagi-Sugeno fuzzy system. We use a new structure of fuzzy RBF neural network, which has been proved that it is better than other structures in term of interpretability [8]. Our model also use a Rival Penalized Competitive Learning (RPCL) and a swarm based algorithm called QPSO to determine design parameters of hidden layer (such as the number of hidden neurons, setting parameters of activation functions and so on) and design parameters of output layer (the weight of output layer), respectively.

RPCL is the best clustering algorithm that is introduced by Li *et al*. [8]. RPCL has the ability of automatically allocating an appropriate number of units for an input data set. RPCL clustering [17] can be regarded as an unsupervised extension of Kohonen's supervised learning vector quantization algorithm LVQ2 [9].

Recently, a novel evolutionary technique, Quantum-behaved Particle Swarm Optimization (QPSO), has been introduced [10, 11, 12]. QPSO is

proposed by combining the classical Particle Swarm Optimization (PSO) philosophy and quantum mechanics to improve performance of PSO [10]. It has been shown that QPSO outperforms original PSO considerably on several widely known benchmark functions [10, 11, 12]. In QPSO, the only setting parameter is contraction expansion coefficient, which is gradually decreased with the number of iterations. Simulation results of nonlinear function approximation have shown that the proposed fuzzy RBF networks have good results.

The paper is organized as follows. RBF networks, fuzzy systems and rival penalized competitive learning are reviewed in section 2. In section 3, quantum behaved particle swarm optimization is reviewed. In section 4, the proposed approach is discussed in details. Some experimental studies are presented in section 5. Finally, we draw some conclusions in section 6.

2. RBF Networks, Fuzzy Systems and Rival Penalized Competitive Learning

2.1. Radial Basis Function Neural Networks

Radial basis function neural networks are one of the most important models of artificial neural networks. They were proposed in [9, 10] among others in the context of different research motivations. Generally, a RBF network with a single output can be expressed as follows:

$$y = \sum_{j=1}^{N} w_j \varphi_j \left(\frac{\| x - c_j \|}{\sigma_j} \right) \tag{1}$$

Where φ_j is called the j^{th} radial-basis function or the j^{th} receptive field unit, c_j and σ_j are the center and the variance vectors of the j^{th} basis function, and w_j is the weight or strength of the j^{th} receptive field unit. If the basis functions of the RBF network are gaussian functions and the output is normalized, an RBF network can be described as:

$$y = \frac{\Sigma_{j=1}^{N} W_j \Pi_{i=1}^{m_j} \exp\left[-\left(\frac{x_i - c_{ij}}{\sigma_{ij}} \right)^2 \right]}{\Sigma_{j=1}^{N} \Pi_{i=1}^{m_j} \exp\left[-\left(\frac{x_i - c_{ij}}{\sigma_{ij}} \right)^2 \right]} \tag{2}$$

Where $1 \le m_j \le M$ is the dimension of the j^{th} basis function, M is the dimension of the input space, and N is the number of hidden nodes. Figure 1 shows an RBF network.

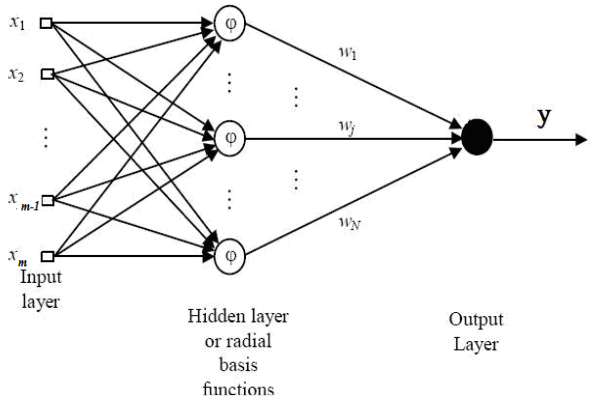


Figure 1. RBF network with p input variables.

2.2. Fuzzy Systems

The theory of fuzzy sets and fuzzy inference systems originated from a completely different research field [18]. Fuzzy inference systems are composed of a set of if-then rules. A Sugeno-Takagi fuzzy model has the following form of fuzzy rules [15]:

$$R_j : \text{IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \text{ and } \ldots \text{ and } x_M \text{ is } A_{Mj}, \text{ Then } y = g_j(x_1, x_2, \ldots, x_M), \tag{3}$$

Where $g_i(.)$ is a crisp function of x_i. The overall output of the fuzzy model can be obtained by:

$$y = \frac{\Sigma_{j=1}^{N} [g_j(.) T_{i=1}^{m_j} \varphi_{ij}(x_i)]}{\Sigma_{j=1}^{N} [T_{i=1}^{m_j} \varphi_{ij}(x_i)]} \tag{4}$$

Where $1 \le m_j \le M$, is the number of input variables that appear in the rule premise, M is the number of inputs, φ_{ij} is the membership function for fuzzy set A_{ij} and T is a t-norm for fuzzy conjunction. It is noticed that the RBF network expressed in equation 2. And the fuzzy systems described by equation 4 are mathematically equivalent provided that multiplication is used for the t-norm in fuzzy systems. Both systems use gaussian basis functions. Figure 2 shows a fuzzy system with two input variables.

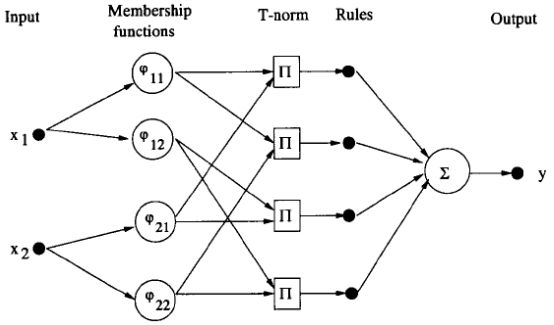


Figure 2. Fuzzy system with two input variables.

2.3. Rival Penalized Competitive Learning

In most of the clustering algorithms, the number of clusters must be given in advance. However it's hard

do so without prior knowledge. RPCL has the ability of automatically allocating an appropriate number of clusters for an input data set. The basic idea is that for each input not only the winner unit is modified to adapt to the input, but also its rival the sub winner is delearned by a smaller learning rate. Assuming there are *k* cluster centers, the cluster center for the winner's unit is accentuated whereas the weight for the second winner, or the rival, is attenuated. The remaining *k-2* centers are unaffected. The winner is defined as the cluster center that is closest to the randomly selected feature vector [17].

RPCL clustering [17] can be regarded as an unsupervised extension of Kohonen's supervised learning vector quantization algorithm LVQ2. It can also be regarded as a variation to the more typical Competitive Learning (CL) algorithms [17]. RPCL is a stochastic clustering algorithm that is able to perform adaptive clustering efficiently and quickly leading to an approximation of clusters that are statistically adequate.

3. Quantum-Behaved Particle Swarm Optimization

3.1. Particle Swarm Optimization

The PSO algorithm, firstly proposed by Kennedy and Eberhart [6], is a population-based evolutionary search technique. It is underlying motivation for the development of PSO was social behaviour of animals such as bird flocking, fish schooling, and animal herding and swarm theory. In PSO with *M* individuals, a potential solution to a problem is represented as a particle flying in *D* dimensional search space, with the position vector $Xi=(x_{i1}, x_{i2}, \ldots, x_{iD})$ and velocity $Vi=(v_{i1}, v_{i2}, \ldots, v_{iD})$. Each particle records its best previous position (the position giving the best fitness value) as $pbesti=(pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$ called personal best position. At each iteration, each particle competes with the others in the neighborhood or in the whole population for the best particle (with best fitness value among neighborhood or the population) with best position $gbesti=(gbest_{i1}, gbest_{i2}, \ldots, gbest_{iD})$ called global best position, and then makes stochastic adjustment according to the following evolution equations.

$$v_{id} = w.v_{id} + c_1.rand().(pbest_{id} - x_{id}) + c_2.rand().(gbest - x_{id}) \quad (5)$$

$$x_{id} = x_{id} + v_{id} \quad (6)$$

For $i=1, 2, \ldots, M$; $d=1, 2, \ldots, D$. In the above equations, c_1 and c_2 are positive constant; rand() and rand() are two random functions generating uniformly distributed random numbers within [0, 1]. Parameter *w* is the inertia weight introduced to accelerate the convergence speed of the PSO. At each iteration, the value of V_{id} is restricted in [-*Vmax*, *Vmax*].

3.2. Quantum-Behaved Particle Swarm Optimization

PSO is not a global convergence-guaranteed optimization algorithm, as Wang and Mendel has demonstrated [16]. Therefore, Sun *et al.* [10, 11] proposed a global convergence-guaranteed search technique QPSO, whose performance is superior to the PSO.

In the quantum model of a PSO, the state of a particle is depicted by wave function ψ(x, t), instead of position and velocity. The dynamic behaviour of the particle is widely different from that of the particle in traditional PSO systems in that the exact values of position and velocity cannot be determined simultaneously. We can only learn the probability of the particle's appearing in position *x* from probability density function $|\Psi(x,t)|^2$, the form of which depends on the potential field the particle lies in. The particles move according to the following iterative equation:

$$x_{id}(t+1) = \left\{ g_{id} \pm \beta \mid mbest_d - x_{id}(t) \mid \ln(1/u) \right\} \quad (7)$$

Where

$$g_{id} = \varphi.pbest_{id} + (1-\varphi)gbest_d \quad (8)$$

and

$$mbest_d = \sum_{i=1}^{m} pbest_{id} / m \quad (9)$$

mbest (mean best position or mainstream thought point) is defined as the mean value of all particles' the best position, φ and *u* are random number distributed uniformly on [0, 1] respectively and *m* is the number of particles. $L = \beta.|mbest_d - x_{id}(t)|.\ln(1/u)$ can be viewed as the strength of creativity or imagination because it characterizes the knowledge seeking scope of the particle, and therefore the larger the value *L*, the more likely the particle find out new knowledge. The parameter β called contraction-expansion coefficient, is the only parameter in QPSO algorithm. From the results of stochastic simulations, QPSO has relatively better performance by varying the value of β from 1.0 at the beginning of the search to 0.5 at the end of the search to balance the exploration and exploitation [16].

4. The Proposed Model of the Fuzzy Neural Network

It has been proved that RBFN is functional equivalent with a simplified class of fuzzy inference systems [2]. However, the only difference between the two systems is interpretability, which makes fuzzy systems easy to understand [4]. Representing a fuzzy system with a general RBFN weakens the outstanding interpretability of fuzzy systems. Therefore, most RBFNs used to

implement fuzzy inference are the transformed ones of the whole linkage form in practice. Those transformed RBFNs can improve the interpretability of networks used to express fuzzy systems. From the analysis and comparison of well-known structures, it can be observed that some structures are simple and capable of describing clearly the fuzzy partitions of input space whereas other structures are not simple and clear but they have better representational power on the interpretability of fuzzy systems.

Li and Hori [8] have proposed a new structure of RBFN for the purpose of improving the interpretability of fuzzy system and simplicity. Figure 4 gives the new network structure, which integrates the natures of well-known networks that are proposed so far. The new structure can not only represent the fuzzy partitions of input space clearly but can also give the formal description of fuzzy systems intuitively. It is proved that the representational power of the new structure is improved greatly, and the ability to clearly express fuzzy partitions is maintained for more details [8].

According to the network structure, the learning algorithm is composed of a fuzzy partition algorithm of input space, a fuzzy inference algorithm which are given in following subsections.

4.1. Fuzzy Partition Part

Let $x=(x_1, x_2, ..., x_N)$ denote the N-dimensional input space, where x_i $i=1, 2, ..., N$ is an input variable; and Y denote the single-dimensional output space. The input layer and hidden layer 1 of the network form the fuzzy partition part. Each input node x_i is connected to the corresponding s_i nodes in hidden layer 1, and s_i denotes the number of fuzzy partition for variable x_i. In this paper, $s=(s_1, s_2, ..., s_N)$ denotes the number of all the fuzzy partitions, notation c_{ik_i} $k_i=1, 2, \ldots, s_i$ denotes the weights from the i^{th} input node to the k_i^{th} node in hidden layer 1, and k_i denotes the k_i^{th} fuzzy partition of the i^{th} input variable. The fuzzy partition labels of N input variables are denoted by notation $k=(k_1, k_2, ..., k_N)$, where each k_i corresponds to $s_i \in (s_1, s_2, ..., s_N)$. The gaussian function is used as the transfer functions of the nodes in hidden layer 1; hence, the output of the nodes can be written as:

$$f_{i,k_i}(x_i) = e^{-(x_i - c_{ik_i})^2 / \sigma_{i,k_i}^2} \tag{10}$$

Where c_{ik_i} and σ_{ik_i} are the center and width of the gaussian function, respectively. In order to determine parameter σ_{ik_i}, a conception of overlap degree is introduced [8]. The overlap degree is the degree by which two fuzzy subsets overlap. In fuzzy control, overlap degree is an important factor that affects control performance. Generally, overlap degree should be around 0.5; a value that is too big or too small may result in an unexpected control effect [8].

Equation 11 gives the mathematical description of nonsymmetrical membership function. As long as membership functions are defined by equation 11 and the selection of widths σ_{il} and σ_{ir} are defined by equations 12 and 13 it can be ensured that the overlap degree of two adjacent fuzzy subsets will be kept about 0.5.

$$f_{i,k_i}(x_i) = \begin{cases} e^{-(x_i - c_{ik_i})^2 / \sigma_{il,k_i}^2}, & x_i \le c_{i,k_i} \\ e^{-(x_i - c_{ik_i})^2 / \sigma_{ir,k_i}^2}, & x_i > c_{i,k_i} \end{cases} \tag{11}$$

$$\sigma_{il,k_i} = \frac{\| c_{i,k_i} - c_{i-1,k_i} \|}{\gamma}, \quad 1 < \gamma < 2 \tag{12}$$

$$\sigma_{ir,k_i} = \frac{\| c_{i+1,k_i} - c_{i,k_i} \|}{\gamma}, \quad 1 < \gamma < 2 \tag{13}$$

Where γ is the overlap coefficient.

Figure 3 shows a fuzzy partition example that the membership functions are described by equation 11.

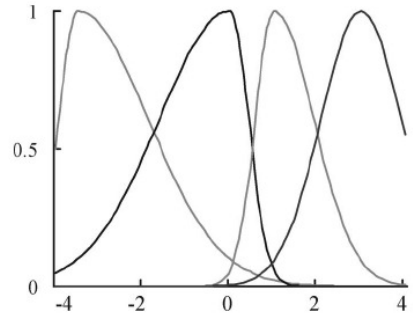


Figure 3. Fuzzy partition when membership functions are described by equation 11.

4.2. Fuzzy Inference Part

The inference task of fuzzy systems is implemented by hidden layer 2 and hidden layer 3. The L node groups denote L rules. In each group, there are N input nodes $P^l{}_{ik_i}$, $i=1, 2, ..., N$ $k_i \in (1, 2, ..., s_i)$, which correspond to the N premises of the l^{th} rule. Notation $P^l{}_{ik_i}$ denotes that the i^{th} premise of the l^{th} rule takes the ki^{th} linguistic value $A^l{}_{ik}$. When there is a crisp input $x^0(x^0{}_1, x^0{}_2, ..., x^0{}_N)$, $P^l{}_{ik_i}$ should be the membership degree of $x^0{}_i$ that belongs to $A^l{}_{ik_i}$. In other words, for l outputs in hidden layer 2 $P^l{}_{ik_i}$ is equal to $f_{i,\,ki}(x^0{}_i)$. The transfer function of each node R_l in hidden layer 3 is product operation. Therefore, the output of node R_l can be given by equation 14:

$$R_l = \prod_{i=1}^{N} P^l_{ik_i} \tag{14}$$

The output of system y is composed of the L consequents of a fuzzy rule. Using weight v_l *between*

the inference layer and the output layer. The output of the network is represented by equation 15.

$$y = \frac{\sum_{l=1}^{L} v_l R_l}{\sum_{l=1}^{L} R_l} = \frac{\sum_{l=1}^{L} v_l \prod_{i=1}^{N} P_{ik_i}^{l}}{\sum_{l=1}^{L} \prod_{i=1}^{N} P_{ik_i}^{l}} \quad (15)$$

4.3. Learning Algorithm of Fuzzy Neural Network

Based on the viewpoint of functional equivalence, the center value c_{ik_i} the width of the gaussian function σ_{ik_i}, and the weight value v_l denote just the three key parameters of the fuzzy system. These parameters need to be modified via learning of the network. It is very useful for the fuzzy RBF network that the structuring and modifying of the fuzzy inference system parameters can be accomplished by learning of the network. The learning algorithm of the fuzzy neural network consists of two parts: the method of structuring the network center c_{ik_i} fuzzy partition algorithm and the learning algorithm of the network weight v_l fuzzy inference algorithm.

In this paper, the RPCL is used to determine the network center c_{ik_i} and the QPSO is used to adjust the network weight v_l.

4.3.1. Fuzzy Partition Algorithm of Input Space

In this paper we use the RPCL clustering algorithm to determine the network center c_{ik_i} and also the equations 12 and 13 to determine the width of the gaussian functions σ_{ik_i}.

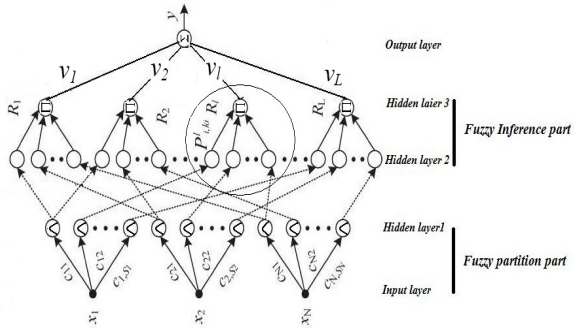


Figure 4. The new fuzzy RBF neural network.

Then we use equation 11 to determine $P^{l}_{ik_i}$, which is the membership degree of the input variable x_i that belongs to $A^{l}_{ik_i}$. The detailed process of the RPCL is as follows:

1. *Step 0:* Initialization randomly pick c_i: i=1, 2, …, k as the initial cluster centers.
2. *Step 1:* Winner-take-all rule randomly take a feature vector x from the feature sample set X, and for i=1, 2, …, k, we let:

$$u_i = \begin{cases} 1, \text{if i = w such that } \gamma_w = \| x - c_w \|^2 = \min_j \gamma_j \| x - c_j \|^2 \\ -1, \text{if i = r such that } \gamma_r = \| x - c_r \|^2 = \min_j \gamma_j \| x - c_j \|^2 \\ 0, \text{otherwise} \end{cases} \quad (16)$$

   Where k is number of samples and w is the winner index, r is the second winner rival index, $\gamma_j = n_j / \sum_{i=1}^{k} n_i$ and n_i is the cumulative number of the occurrences of u_i=1. This term is added to ensure that every cluster center will eventually become the winner during the updating process.
3. *Step 2:* Updating cluster centers update the cluster center vector c_i by equation 17:

$$\Delta c_i = \begin{cases} \alpha_w (x - c_i), & \text{if } u_i = 1 \\ -\alpha_r (x - c_i), & \text{if } u_i = -1 \\ 0, & \text{otherwise} \end{cases} \quad (17)$$

   Where $0 \le \alpha_w, \alpha_r \le 1,$ are the learning rates for the winner and rival unit, respectively.

Since we use equations 11, 12 and 13 to determine the width of the gaussian functions and the membership degree of the input variable x_i, we guarantee that the overlap degree of the adjacent fuzzy subsets is kept about 0.5.

4.3.2. Fuzzy Inference Algorithm

In order to adjust the network weight v_l by QPSO, the fitness function and the encoding of individuals (also known as particle position) must be defined. So, we define particle position as shown in equation 18:

$$V = [v_1, v_2, ..., v_l, ..., v_L] \quad (18)$$

Where v_l is weight between the inference layer and the output layer and L is number of nodes in inference layer.

For the M sample data (X_h, y_h) h=1, 2, 3, ..., M. we consider o_h as the desired output of the network, y_h as network output and $E = \frac{1}{2}\sum_{h=1}^{M}(y_h - o_h)^2$ as the target error function. The performance of each particle is evaluated according to its fitness. The fitness function is defined as follows:

$$fitness = \frac{1}{1 + \frac{1}{2M}\sum_{h=1}^{M}(y_h - o_h)^2} \quad (19)$$

Where M is number of training samples, y_h, o_h are the network output and desired output for sample h respectively. The following is the procedure of QPSO for learning weights:

Step 0: Initialize the population by randomly generate the position vector V_i of each particle and set $pbest_i = V_i$.

2. *Step 1:* Evaluate the fitness value of each particle by equation 19, update the personal best position $pbest_i$ and obtain the global best position *gbest* across the population.
3. *Step 2:* If the stop criterion is met, go to step 4; or else go to step 3.
4. *Step 3:* Update the position vector of each particle according to equation 7 and go to step 1.
5. *Step 4:* Output the *gbest* as optimized parameters.

In QPSO, the only setting parameter is contraction-expansion coefficient β, which is gradually decreased for the interval [1.2, 0.5] with the number of iterations.

There are two alternatives for stop criterion of the algorithm. One method is that the algorithm stops when $E=\frac{1}{2}\Sigma_{h=1}^{M}(y_h-o_h)^2$ is less than a given threshold ε; the other is that it terminates after executing a pre-specified number of iterations.

5. Computer Simulation

To determine the efficiency of the algorithm, we run experiments on a general purpose PC (CPU: 1.2GHZ Pentium III; Memory: 128MB; OS: Windows XP). The algorithm is programmed in Java language. The initial populations consist of random particles and each of the experiments was repeated 30 runs. In our proposed algorithm, the only setting parameter is contraction-expansion coefficient β, which was gradually decreased for the interval [1.2, 0.5] with the number of iterations. To demonstrate the effectiveness of the proposed algorithm, there are two examples problems. The results of this proposed algorithm are compared with gradient base method.

Example 1: Two-input nonlinear sinc function. The function is defined as:

$$z=\frac{\sin(x)\sin(y)}{xy},\ x,y\in[-10,10] \tag{20}$$

The training data consisted of uniformly sampled 400 two-input data and the corresponding target data. Another set of uniformly sampled 100 input-target data was used as the testing data.

Figure 5 shows the Root Mean Square Error (RMSE) curve of the fuzzy neural network trained by the QPSO and gradient-based methods with learning rate 0.02, after 300 iterations of learning, respectively. For the M sample data (X_h, y_h) h=1, 2, 3, …, M. The RMSE is calculated as follow:

$$RSME=\sqrt{\frac{1}{2M}\Sigma_{h=1}^{M}(y_h-o_h)^2} \tag{21}$$

Where o_h is a desired output of the network and y_h is a network output.

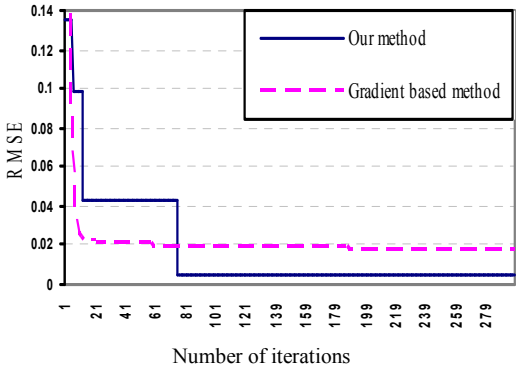


Figure 5. RSME during training.

The RMSE of training of proposed method is 0.00532 and the RMSE of testing is 0.00956 whereas the RMSE of training of gradient based method is 0.01737 and the RMSE of testing is 0.01903. Figure 5 shows that the proposed algorithm is better than gradient base method in term of RMSE. Figures 6 and 7 show the error curve of the fuzzy neural network during training and testing, respectively.

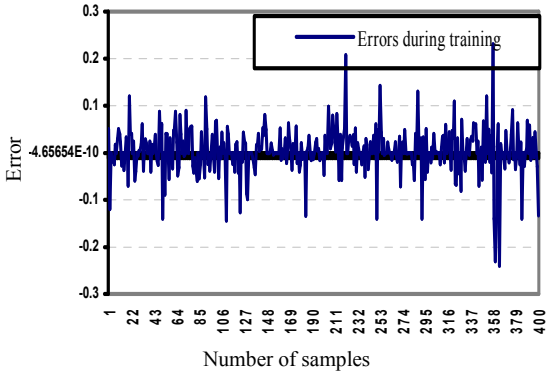


Figure 6. Errors during training.

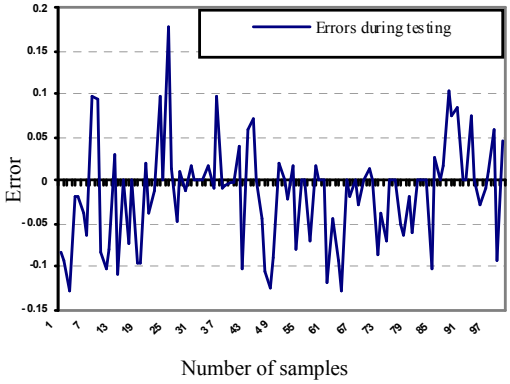
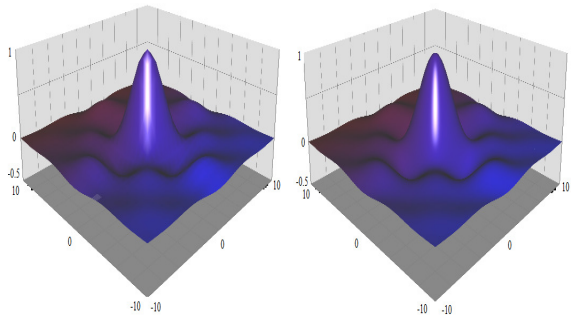


Figure 7. Errors during testing.

In order to test the learning effect and verify the generalized ability of the network, a set of new data different from the trained pattern data is used as the input to the network which was trained. The simulation result of the generalization ability of the proposed network as shown in Figure 8.



a) The desired output. b) The actual output.

Figure 8. Model of the fuzzy neural network.

Example 2: Two-input nonlinear function. The function is defined as:

$$z = sin(x^2)sin(y^2), \; x,y \in [-2,2] \qquad (22)$$

The training data consisted of uniformly sampled 400 two-input data and the corresponding target data. Another set of uniformly sampled 100 input-target data was used as the testing data. The results are presented in Figures 9 and 12.

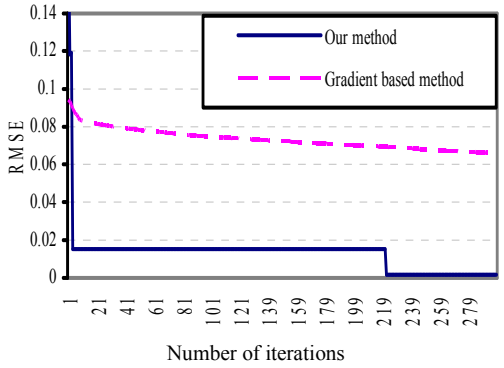


Figure 9. RMSE during training.

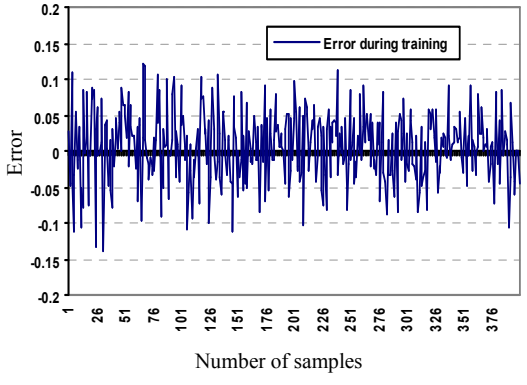


Figure 10. Errors during training.

The RMSE of training is 0.00152 and the RMSE of testing is 0.0103 whereas the RMSE of training of gradient based method is 0.06709 and the RMSE of testing is 0.08903. Figure 5 shows that the proposed algorithm is better than gradient base method in term of RMSE.

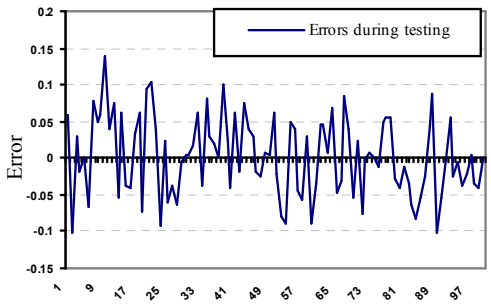
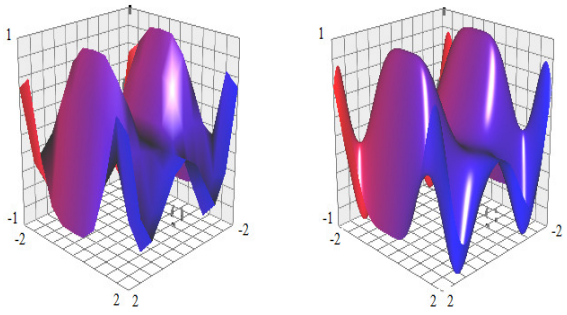


Figure 11. Errors during training.



a) The desired output. b) The actual output.

Figure 12. Model of the fuzzy neural network.

6. Conclusions

In this paper, four-layer fuzzy-neural network structure based on RBF neural network and some algorithms for determining design parameters of hidden layers (such as the number of hidden neurons, setting parameters of activation functions and so on) and design parameters of output layer (the weight of output layer) have been proposed. According to the new structure, we have used nonsymmetric gaussian function as activation function. This function has one center and two widths, which have been described in the paper. To determine the number of neurons and center of the nonsymmetric gaussian function, RPCL has been used. RPCL is the best clustering algorithm that has the ability of automatically allocating an appropriate number of clusters without any knowledge. The widths of the nonsymmetric gaussian function have been calculated by equations 12 and 13. In order to adjust the network weight, QPSO has been used. QPSO is a global convergence-guaranteed optimization algorithm. It not only is better them PSO in term of performance but also has only one setting parameter.

At the end, we have compared the performance of our method with gradient-based method under the same condition. Our method and gradient-base method have been applied to nonlinear function approximation. From the simulated experiment, the result of our method is better than gradient-base method and requires less number of iterations. On the other hand, our method can solve these problems with faster

convergence speed and have more powerful optimizing ability.

References

[1] Bao H., Huang H., and Li X., "A Study of the T-S Fuzzy RBF Neural Network," *Journal of Huazhong University of Science and Technology*, vol. 27, no. 1, pp. 11-13, 1999.

[2] Jang S. and Sun T., "Functional Equivalence between Radial Bases Functions and Fuzzy Inference Systems," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 156-158, 1993.

[3] Jang S. and Sun T., "Neuro-Fuzzy Modeling and Control," *in Proceedings of the IEEE*, pp. 378-406, 1995.

[4] Jin Y. and Sendhoff B., "Extracting Interpretable Fuzzy Rules from RBF Networks," *Neural Processing Letters*, vol. 17, no. 2, pp. 149-164, 2003.

[5] Jin Y., "Fuzzy Modeling of High-Dimensional Systems Complexity Reduction and Interpretability Improvement," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 2, pp. 212-221, 1995.

[6] Kennedy J. and Eberhart C., "Particle Swarm Optimization," *in Proceedings of IEEE International Conference on Neural Network*, Australia, pp. 1942-1948, 1995.

[7] Linkens A. and Chen Y., "Input Selection and Partition Validation for Fuzzy Modeling Using Neural Network," *Journal of Fuzzy Sets Systems*, vol. 107, no. 3, pp. 299-308, 1999.

[8] Li W. and Hori Y., "An Algorithm for Extracting Fuzzy Rules Based on RBF Neural Network," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 4, pp. 1269-1276, 2006.

[9] Rosenblatt R., *Principles of Neurodynamics*, Spartan Books, New York, 1959.

[10] Sun J., Feng B., and Xu B., "Particle Swarm Optimization with Particles Having Quantum Behavior," *in Proceedings of Congress on Evolutionary Computation*, USA, pp. 325-331, 2004.

[11] Sun J. and Xu B., "A Global Search Strategy of Quantum-Behaved Particle Swarm Optimization," *in Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, Singapore, pp. 111-116, 2004.

[12] Sun J. and Xu B., "Adaptive Parameter Control for Quantum-Behaved Particle Swarm Optimization on Individual Level," *in Proceedings of IEEE International Conference on Systems*, USA, pp. 3049-3054, 2005.

[13] Sun J. and Xu W., "Parameter Selection of Quantum-Behaved Particle Swarm Optimization," *in Proceedings of Advances in Natural Computation*, Heidelberg, pp. 543-552, 2005.

[14] Takagi T. and Sugeno M., "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *in Proceedings of IEEE Transactions on Systems*, pp. 116-132, 1985.

[15] Wang X. and Mendel J., "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on Systems*, vol. 22, no. 6, pp. 1414-1427, 1992.

[16] Xu L. and Krzyzak A., "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 636-649, 1993.

[17] Zadeh A., "Outline of A New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems*, vol. 3, no. 1, pp. 28-44, 1973.

Saeed Farzi received his BS in computer engineering from Razi University, Iran in 2004, and his MS in artificial intelligence from Isfehan University, Iran in 2006. He is a faculty member at the Department of Computer Engineering, Islamic Azad University-beranch of Kermanshah, Iran. His current research interests include artificial intelligence, neural network, soft computing and grid computing.