Jian SUN[1], Xiongwei ZHANG[2], Tieyong CAO[2], Jibin YANG[2], Xinjian SUN[1], Qibin LIAO[3]

Inst. Communication Engineering, PLA Univ. of Sci. & Tech.[(1)], Inst. Command Automation, PLA Univ. of Sci.&Tech.[(2)], 75203 PLA Troops[(3)]

A Novel Structure Design and Training Algorithm for Quantum Neural Network

Abstract. *In the structure of original Quantum Neural Network (QNN), only multi-sigmoid transfer function is adopted. Besides that, due to the conflict of the two objective functions in original training algorithm, the training process converges slowly and presents constant variation. In this paper, the QNN with multi-tan-sigmoid transfer function and a novel training algorithm which combines the two objective functions are proposed. Experimental results demonstrate the effectiveness of the structure improvement and the new training algorithm.*

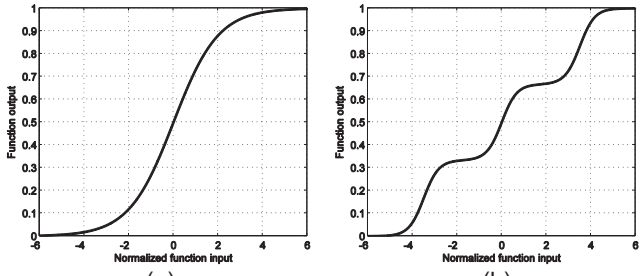
Streszczenie. *W oryginalnym algorytmie kwantowej sieci neuronowej QNN tylko multisigmoidalna funkcja przejścia jest wykorzystywana. W pracy zaprezentowano sieć z multi-tan-sigmoidalną funkcją przejścia z nowym algorytmem uczenia. (**Nowa struktura i algorytm uczenia w kwantowej sieci neuronowej**)*

Keywords: Quantum Neural Network; transfer function; multi-tan-sigmoid; training algorithm.
Słowa kluczowe: kwantowa sieć neuronowa, algorytm uczenia.

1. Introduction

Artificial Neural Network (ANN) is a power parallel processing tool inspired by the real brain [1]. The computation capability of ANN lies in its complex interconnected group of artificial neurons. Taking advantage of this computation capability, ANN has been widely used in a variety of areas, such as data fitting, pattern recognition and data mining [1]. However, traditional ANN doesn't have the ability of identifying fuzziness in the data which limits its usage in pattern recognition.

In order to cope with this problem, Purushothaman and Karayiannis proposed a new kind of ANN: the Quantum Neural Network (QNN) [2]. To acquire the ability of identifying uncertainty in data, QNN forms graded boundaries of the feature space instead of the sharp boundaries between different patterns. The graded boundaries are obtained by replacing the single sigmoid transfer function in each neuron with the superposition of several sigmoid functions. This superposition generates multistep transfer function which is shown in Fig 1. In this figure, the multi-sigmoid function is constructed by the superposition of 3 sigmoid functions with equal shift intervals.



(a) (b)

Fig.1. The original sigmoid transfer function (a) and the multi-sigmoid transfer function (b)

Due to the fuzziness identification ability, QNN has been widely applied in different areas, such as the classification of power quality disturbances [3], ammonia identification [4] and TP film printing [5].

However in the work of Purushothaman and Karayiannis, only sigmoid function is used to form the multi-step transfer function in QNN. Other kinds of widely used transfer function are not considered, such as tan-sigmoid function. In this paper, QNN with the tan-sigmoid formed transfer function is introduced. In additional to the network structure modification, the original training algorithm is also needed to be improved. More specifically, there are two objective functions in the original training algorithm which are used to update two sets of parameters in QNN. As the objective functions are independently minimized, conflict may frequently appear which means minimizing one objective function often causes the expansion of another objective function.

In this paper, a training algorithm considering the conflict of the two objective functions is proposed. Through combining the two objective functions, the conflict between them can be removed which leads to faster and more stable training of QNN.

The paper is organized as follows. In the next section, the QNN is briefly reviewed and the multi-tan-sigmoid transfer function is introduced. In section 3 the new training algorithm is proposed. In section 4, experiments are carried out to analyse the new QNN structure and verify the validity of new training algorithm. Finally, the conclusions and discussions are summarized in section 5.

2. QNN and the new transfer function

QNN is feedforward neural network with only one hidden layer. Compared with traditional feedforward neural network, the only difference between them is that the QNN adopts multi-sigmoid transfer function. A QNN with n_i input, n_h hidden neurons and n_o output neurons is shown

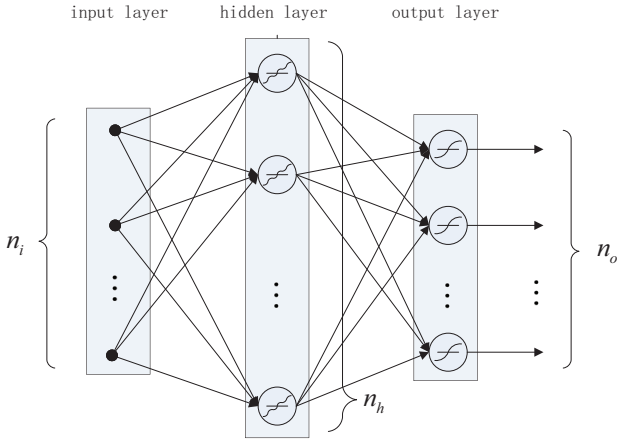


Fig.2. The structure of QNN

Let the inputs and outputs of QNN be $\mathbf{X}$ and $\mathbf{Y}$ respectively:

(1)
$$\mathbf{X}=\left[\mathbf{x}_1,\mathbf{x}_2,\cdots\mathbf{x}_k,\cdots,\mathbf{x}_K\right]$$
$$\mathbf{Y}=\left[\mathbf{y}_1,\mathbf{y}_2,\cdots,\mathbf{y}_k,\cdots,\mathbf{y}_K\right]$$

where

(2)
$$\mathbf{x}_k=\left[x_{1,k},x_{2,k},\cdots,x_{n_i,k}\right]^T$$
$$\mathbf{y}_k=\left[y_{1,k},y_{2,k},\cdots,y_{n_o,k}\right]^T$$

and K is the number of inputs and outputs vector pairs.

The multi-sigmoid transfer function can be presented as:

(3)
$$\tilde{h}_{j,k}=\frac{1}{n_s}\sum_{r=1}^{n_s}h_{j,k}^r=\frac{1}{n_s}\sum_{r=1}^{n_s}\mathrm{sgm}\left(\beta_h\left(\bar{h}_{j,k}-\theta_j^r\right)\right)$$

where $\tilde{h}_{j,k}$ is output of the j th hidden neuron when the input is $\mathbf{x}_k$. $h_{j,k}^r$ is the output of the r th sigmoid function in the j th hidden neuron and $\bar{h}_{j,k}$ is its input. n_s is the number of superimposed sigmoid function which is also called the interval number of transfer function. And θ_j^r is the shift value of the r th sigmoid function which is also called quantum interval. β_h is slope parameter in the sigmoid function. $\mathrm{sgm}(\tau)$ represents the sigmoid function as shown in equation (4):

(4)
$$\mathrm{sgm}(\tau)=\frac{1}{1+\exp(-\tau)}$$

In addition the input of the j th transfer function $\bar{h}_{j,k}$ is given by equation (5):

(5)
$$\bar{h}_{j,k}=\sum_{i=1}^{n_i}\upsilon_{ji}x_{i,k}$$

where υ_{ji} is the weight between the i th input node and the j th hidden node.

The output of QNN is obtained by the following equation:

(6)
$$\hat{y}_{i,k}=\begin{cases}\mathrm{sgm}\left(\beta_o\left(\bar{y}_{i,k}\right)\right) & \text{If the output unit is sigmoid}\\ \bar{y}_{i,k} & \text{If the output unit is linear}\end{cases}$$

where $\bar{y}_{i,k}$ is the input of the i th output nodes which can be represented by equation (7). And $\hat{y}_{i,k}$ is the i th output of QNN for input $\mathbf{x}_k$. β_o is the slope parameter of the sigmoid function in output layer. Usually β_o is set to 1.

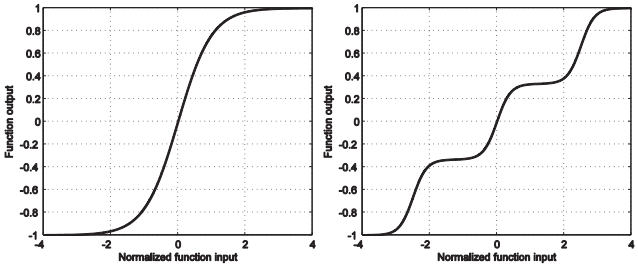
(7)
$$\bar{y}_{i,k}=\sum_{j=1}^{n_h}\omega_{ij}\tilde{h}_{j,k}$$

where ω_{ij} is the weight between the j th hidden node and the i th output node.

The transfer function of QNN discussed above is based on sigmoid function which is shown in Fig. 1. This function can generate outputs between 0 and 1. Besides the sigmoid function, there is another frequently used transfer function: tan-sigmoid function, which is given in equation (8):

(8)
$$\mathrm{tansgm}(\tau)=\frac{2}{1+\exp(-2\tau)}-1$$

The output of tan-sigmoid function is between -1 and 1 which is larger than that of sigmoid function. Through superimposing the tan-sigmoid functions, the graded transfer function can also be achieved and used in QNN. The tan-sigmoid function and the multi-tan-sigmoid transfer function are shown in Fig. 3.



(a) (b)

Fig.3. The tan-sigmoid function (a) and the multi-tan-sigmoid transfer function (b)

By adopting the multi-tan-sigmoid transfer function, a new QNN structure is introduced. And the performance of this new structure will be analyzed by experiment in section 4. Due to the change in transfer function, its training algorithm also needs to be updated. Instead of just updating the training algorithm according to the transfer function, a fundamental improvement of the training algorithm will be presented in the next section.

3. The training algorithm for QNN

3.1 The original training algorithm and its shortage

There are two sets of parameters in the QNN: the weights between layers and the quantum intervals of the transfer function in hidden layer. The weights parameters are $\{\upsilon_{ji}\}$ and $\{\omega_{ij}\}$. Accordingly the training process consists of the adapting these two sets of parameters.

In the original training algorithm, these two sets of parameters are alternately updated according to two objective functions. More specifically, the QNN weights are adapted toward minimizing the squared error of QNN outputs which is denoted by E in equation (9). And the quantum intervals are updated by minimizing the class-conditional variances at the outputs of the hidden units which is denoted by G in equation (10).

(9)
$$\{\upsilon_{ji},\omega_{ij}\}=\arg\min\left\{E=\frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n_o}\left(y_{i,k}-\hat{y}_{i,k}\right)^2\right\}$$

(10)
$$\{\theta_j^r\}=\arg\min\left\{G=\frac{1}{2}\sum_{j=1}^{n_h}\sum_{m=1}^{n_o}\sigma_{j,m}^2\right\}$$
$$=\arg\min\left\{\frac{1}{2}\sum_{j=1}^{n_h}\sum_{m=1}^{n_o}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m}\left(\left\langle\tilde{h}_{j,C_m}\right\rangle-\tilde{h}_{j,k}\right)^2\right\}$$

where C_m is the m th class. And $\left\langle\tilde{h}_{j,C_m}\right\rangle$ is the output average of the j th node in hidden layer when $\mathbf{x}_k\in C_m$ which is obtained by equation (11):

(11)
$$\left\langle\tilde{h}_{j,C_m}\right\rangle=\frac{1}{|C_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m}\tilde{h}_{j,k}$$

where $|C_m|$ denotes the cardinality of C_m.

When QNN is used in pattern recognition, the number of output nodes in QNN is equal to the class number of input

data. In the ideal case, when the input data belongs to the i th class, the i th output node of QNN should be 1 and all other nodes should be 0.

In the original training process, the weights and quantum intervals are alternately updated according to the objective function (9) and (10) by gradient descent method in an iterative way. However as weights and quantum intervals are trained independently, the side effect on G has not been considered when weights are adapted towards the minimization of E, vice versa. Commonly the changes on weights can reduce the value of E but enlarge the value of G. So are the changes on quantum intervals. This conflict on E and G will increase the training time and causes constant fluctuation on the E and G in the training process.

Through experimental results, it can be found that at the beginning the training process, the E and G are decreased synchronously. This means the two objective functions have not conflicted with each other at this phase. As the training iterative increasing, the conflict between E and G become more and more severe.

3.2 New training algorithm for QNN

In order to eliminate the conflict between the two objective functions, the adaptation process of QNN parameters should consider their effect on E and G simultaneously.

As the training process is accomplished by gradient descent algorithm, solving the conflict problem should be by adding non-enlarging constraint in each gradient descent iterative which is presented in the following equations:

$$\left\{\upsilon'_{ji},\omega'_{ij}\right\}=f\left(\left\{\upsilon_{ji},\omega_{ij}\right\}\right) \quad \text{s.t.}\quad G\left(\left\{\upsilon'_{ji},\omega'_{ij}\right\}\right)-G\left(\left\{\upsilon_{ji},\omega_{ij}\right\}\right)\le 0 \tag{12}$$

$$\left\{\theta'^{r}_{j}\right\}=g\left(\left\{\theta^{r}_{j}\right\}\right) \quad \text{s.t.}\quad E\left(\left\{\theta'^{r}_{j}\right\}\right)-E\left(\left\{\theta^{r}_{j}\right\}\right)\le 0 \tag{13}$$

where $f(\cdot)$ and $g(\cdot)$ represents the gradient descent process in each iterative.

However, adding this constraint to the gradient descent algorithm is not a trivial matter. It can be seen from equation (12) and (13) that the constraint inequality functions changes in every iterative. Then this problem cannot be solved directly by the methods in Optimization Theory with inequality constraint.

In this paper, we try to deal with this problem by combining the two objective functions as a whole to update QNN parameters. In addition, the maintenance of original affiliation between the parameter set and objective function will also be considered. Then the following objective functions are introduced:

$$\left\{\upsilon_{ji},\omega_{ij}\right\}=\arg\min\left\{E+\beta_E G\right\} \tag{14}$$

$$\left\{\theta^{r}_{j}\right\}=\arg\min\left\{G+\beta_G E\right\} \tag{15}$$

where β_E and β_G are the non-negative parameters used to adjust the training focus between E and G.

By using the equation (14) and (15) as the objective function, the effects on E and G will all be considered in each QNN parameter adaptation iterative. In the original training algorithm, the QNN weight parameters are trained according to E. In equation (14), it also hopes to focus on the minimization of E so as to maintain this affiliation. Then the parameter β_E is used to adjust this focus. The same is true in equation (15). From the experimental results, we find that difference between the values of E and G are usually significant. To eliminate the effect of value difference on the training, β_E and β_G are obtained by the following equations:

$$\beta_E=\frac{E_p}{2G_p} \tag{16}$$

$$\beta_G=\frac{G_p}{2E_p} \tag{17}$$

where E_p and G_p are the value of E and G before the each training iterative. Dividing the results by 2 is to make sure the training focus is on the first part in each objective function.

Using equation (16) and (17), β_E and β_G can be adjusted adaptively in each iteration. Then according to the new objective functions, the QNN parameters can be trained using the gradient descent algorithm. Due to the changes on transfer function and objective functions, the original training algorithm given in [2] has to be modified. The specific training algorithm is presented in Appendix 1.

4. Experimental results

In order to verify the validity of the new structure and training algorithm, QNN based speaker identification experiments are carried out. TIMIT database [6] is used as the material for the experiments. More specifically, 10 speaker's speech data is randomly selected from the TIMIT database. Then stable voiced segments of the speech data are extracted and divided into frames with the length of 40ms. The overlap rate between adjacent frames is 50%. After that, 10-order Linear Prediction Cepstrum Coefficient (LPCC) and 16-order Mel Frequency Cepstrum Coefficient (MFCC) are calculated for each speech frame. The LPCC and MFCC are combined into a 26-dimension vector used as the input for QNN. According to Zhou et al [7], the combination of LPCC and MFCC will lead to more stable result for the speaker identification.

The output of QNN is a 10-dimention vector which indicates the speaker identity. If the input data belongs to the m th speaker, the m th element of the output vector should be 1 and all other elements should be 0.

250 frames of speech data for each speaker are used as training data and another 250 frames for each speaker are used for testing the training performance.

In these experiments, three QNN are constructed and trained for the speaker identification: the QNN with multi-sigmoid transfer function trained by the original training algorithm, the QNN with multi-tan-sigmoid transfer function trained by the original training algorithm and the QNN with multi-tan-sigmoid transfer function trained by the new training algorithm. All of the three QNNs have 58 nodes in the hidden layer. And the tan-sigmoid transfer functions are used in their output layers.

The changes of E and G in the training process are shown in Fig. 4. It can be known from Fig. 4 that E and G in the new algorithm can reach steady state faster compared with the original algorithm. Besides that, obvious fluctuations of E and G can be observed in the original algorithm. Although the convergence process in (a) and (b) seem fast too, the training results are not satisfactory actually. As can be seen, The value of E in (a) is much higher than that in (c) and (e), which means it has fallen into the local minimum results.

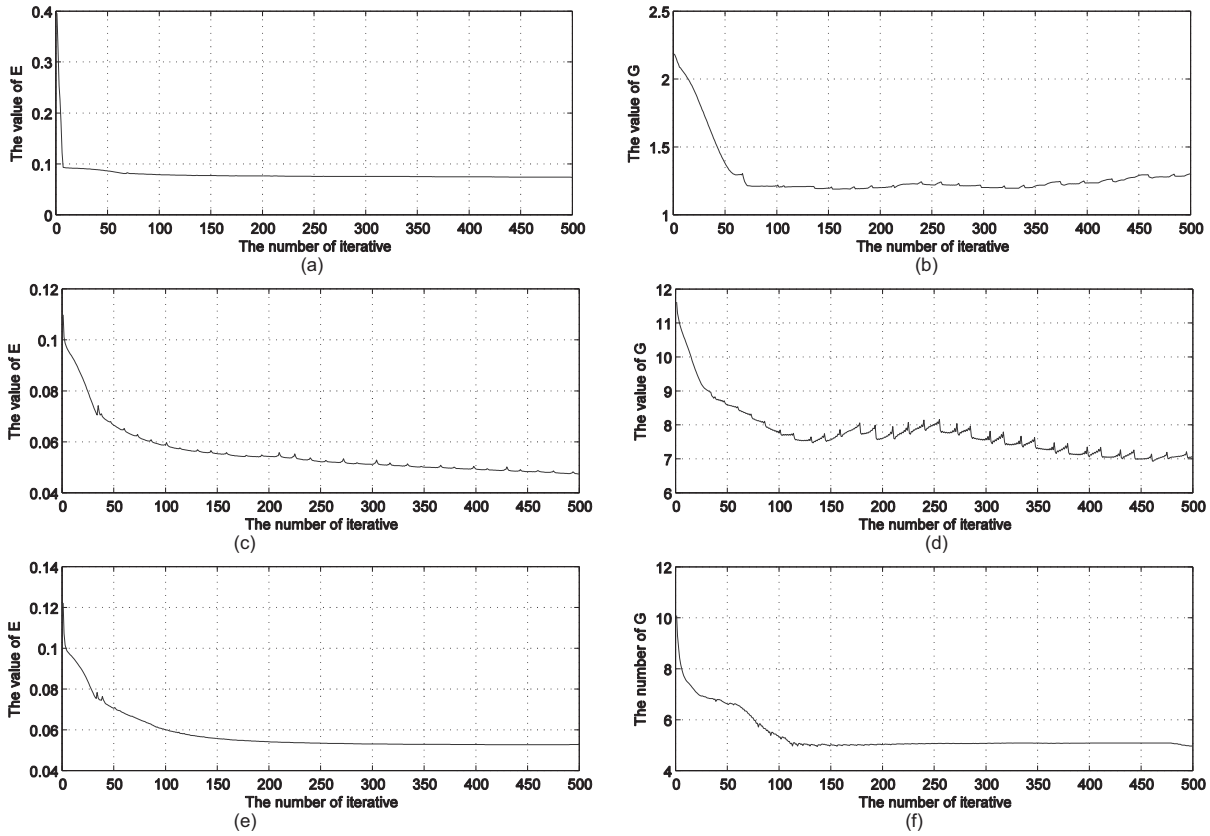


Fig.4. The changes of E and G in the training process. (a) and (b) are the results of multi-sigmoid based QNN trained by original algorithm; (c) and (d) are the results of multi-tan-sigmoid based QNN trained by original algorithm; (e) and (f) are the results of multi-tan-sigmoid based QNN trained by new algorithm.

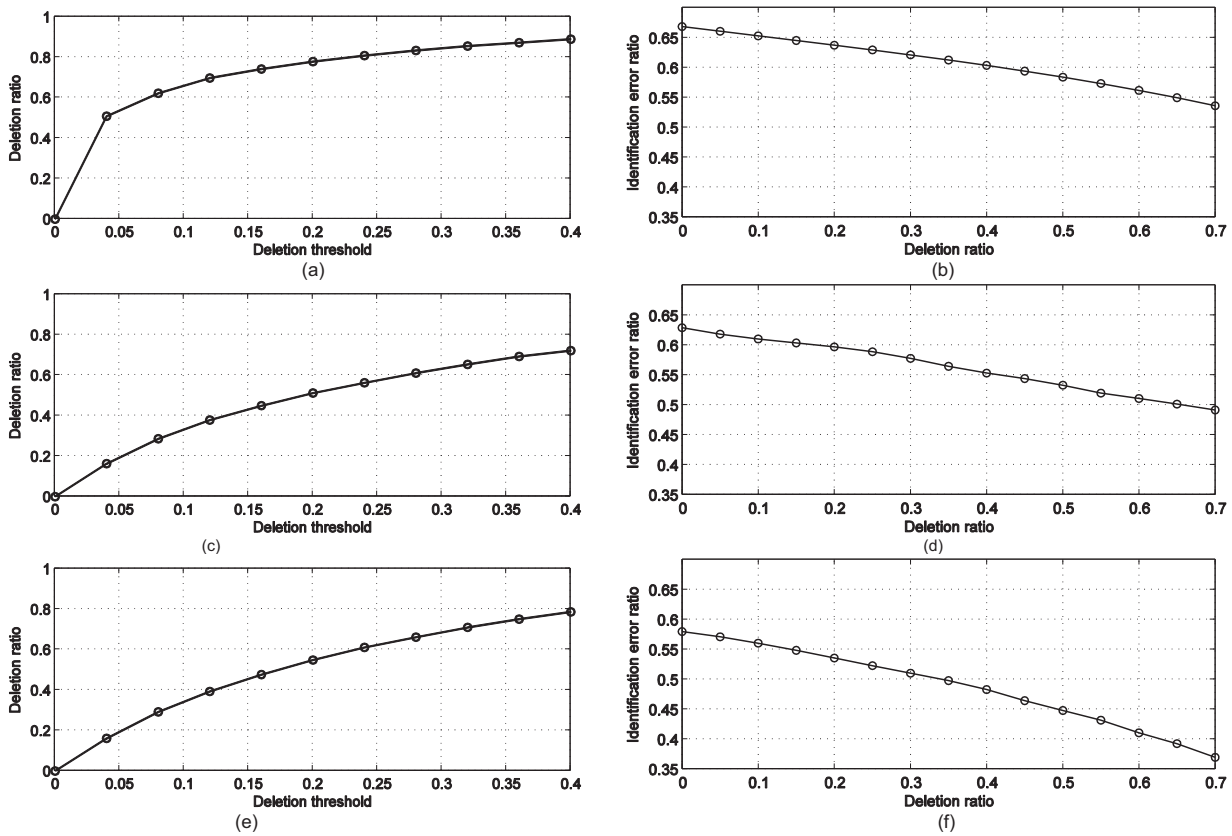


Fig.5. The speaker identification results of the three QNNs. (a) and (b) are the results of multi-sigmoid based QNN trained by original algorithm; (c) and (d) are the results of multi-tan-sigmoid based QNN trained by original algorithm; (e) and (f) are the results of multi-tan-sigmoid based QNN trained by new algorithm.

In the testing phase, the fuzziness data will be deleted based on the output of the QNN in the first place. The deletion is depends on the difference between the two maximum elements in each QNN's output vector. When the difference is smaller than a certain threshold, the input data will be deleted for containing too much fuzziness. Different deletion thresholds are tried in the experiments. And the deletion ratios against the deletion thresholds for the three QNNs are shown in (a), (c), (e) of Fig. 5.

After the fuzziness deletion process, the identification results are obtained based on the remaining data and the QNN outputs. For the three QNNs, the identification error rates for the remaining inputs against the deletion ratios are presented in (b), (d), (f) of Fig. 5. In order to obtain more intuitive results, the data in these three figures is calculated through cubic spline interpolation, as the original deletion ratios in the experiments are not equally spaced along the axis.

It can be known from Fig. 5 that the multi-sigmoid based QNN trained by the original algorithm has the highest deletion ratio almost at every deletion threshold. But the identification performance is the lowest according to the identification error ratios. Obviously the new algorithm trained QNN with multi-tan-sigmoid transfer function outperforms the other two QNNs.

From the experimental results, we know that the new structure of QNN can improve the performance of speaker identification slightly. Compared with the original algorithm, the new training algorithm for QNN can obviously improve its performance.

5. Conclusion and discussion

In this paper, a new transfer function and training algorithm for QNN are proposed. The new training algorithm can eliminate the conflict between the two objective functions in the original training algorithm. Through the experiments results of speaker identification, the validity of the new QNN structure and the training algorithm has been demonstrated.

Although the QNN can handle the uncertainty in data, its pattern recognition performance is not satisfactory. This may attribute to its single hidden layer structure. In the future work, QNN with multi-hidden layers will be our next research direction.

Acknowledgement

The authors wish to acknowledge the financial support of National Natural Science Foundation No.61072042 and the Pre-Research Foundation of PLA Univ. of Sci. & Tech. No. 20110205 and No. 20110211.

REFERENCES

[1] Graupe D., Principles of Artificial Neural Networks. 2nd. *River Edge,* NJ, USA: World Scientific Publishing Co., Inc., (2007).

[2] Purushothaman G., Karayiannis N. B., Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks, *Neural Networks, IEEE Transactions on*, 8(1997), No. 3, 679-693.

[3] He Z., Zhang H., Zhao J., et al., Classification of power quality disturbances using quantum neural network and DS evidence fusion, European Transactions on Electrical Power, (2011).

[4] Shen C., Huang H., Hwang R., Ammonia identification using shear horizontal surface acoustic wave sensor and quantum neural network model, *Sensors and Actuators A: Physical*, 147(2008), No. 2, 464-469.

[5] Huang C., Huang H., Chen Y., et al., An AI system for the decision to control parameters of TP film printing, *Expert Systems with Applications*, 36(2009), No. 5, 9580-9583.

[6] Zue V., Seneff S., Glass J., Speech database development at MIT: Timit and beyond, *Speech Communication*, 9(1990), No. 4, 351-356.

[7] Yuhuan Z., Xiongwei Z., Jinming W., et al., Research on speaker feature dimension reduction based on CCA and PCA, *Wireless Communications and Signal Processing (WCSP)*, 2010 International Conference on, China.

***Authors**:*
Jian Sun, Postgraduate Team 2, Institute of Communications Engineering, Biaoyin 2, Yudao Street, Nanjing, China, 210007, Email: sunjian001@gmail.com;
Xiongwei Zhang, Institute of Command Automation, PLA Univ. of Sci. & Tech.

Appendix 1 The new training algorithm for QNN

1. Select β_h, α, α_θ, where α is the learning step for weights in the gradient descent algorithm and α_θ is the learning step for quantum intervals.

2. Randomly initialize the weights and quantum intervals.

3. Update the weights:

For $k=1,2,\cdots,K$:

For $j=1,2,\cdots,n_h$:

$$h_{j,k} \leftarrow \sum_{i=1}^{n_i} \upsilon_{ji} x_{i,k}$$

For $r=1,2,\cdots n_s$:

$$h_{j,k}^r \leftarrow \mathrm{tansgm}\left(\beta_h\left(\overline{h}_{j,k}-\theta_j^r\right)\right)$$

$$\tilde{h}_{j,k} \leftarrow \frac{1}{n_s}\sum_{r=1}^{n_s} h_{j,k}^r$$

For $i=1,2,\cdots,n_o$:

$$y_{i,k} \leftarrow \sum_{j=1}^{n_h} \omega_{ij}\tilde{h}_{j,k}$$

$$\hat{y}_{i,k} \leftarrow \begin{cases} \mathrm{tansgm}\left(\overline{y}_{i,k}\right) & \text{If the } i\text{th output unit is tansgm} \\ \overline{y}_{i,k} & \text{If the } i\text{th output unit is linear} \end{cases}$$

$$\varepsilon_{i,k}^o \leftarrow \begin{cases} \left(y_{i,k}-\hat{y}_{i,k}\right)\left(1-\hat{y}_{i,k}^{\ 2}\right) & \text{If the } i\text{th output unit is tansgm} \\ y_{i,k}-\hat{y}_{i,k} & \text{If the } i\text{th output unit is linear} \end{cases}$$

For $j=1,2,\cdots,n_h$ and $i=1,2,\cdots,n_o$:

$$\omega_{ij} \leftarrow \omega_{ij} + \alpha\varepsilon_{i,k}^o \tilde{h}_{j,k}$$ ---- updating the weight ω_{ij}

For $j=1,2,\cdots,n_o$:

$$\varepsilon_{j,k}^h \leftarrow \left(\frac{1}{n_s}\sum_{r=1}^{n_s}\left(1-h_{j,k}^{r\ 2}\right)\right)\sum_{i=1}^{n_o}\varepsilon_{i,k}^o\omega_{ij}$$

For $j=1,2,\cdots,n_h$ and $i=1,2,\cdots,n_i$:

$$o_{ji,k} \leftarrow \frac{1}{n_s}\sum_{r=1}^{n_s}\left(1-h_{j,k}^{r\ 2}\right)x_{i,k}$$

$$\upsilon_{ji} \leftarrow \upsilon_{ji} + \alpha\beta_h\varepsilon_{j,k}^h x_{i,k}$$

---- updating the weight υ_{ji} according to E

For $i=1,2,\cdots,n_o$:

$$y_{i,k} \leftarrow \sum_{j=1}^{n_h}\omega_{ij}\tilde{h}_{j,k}$$

$$\hat{y}_{i,k} \leftarrow \begin{cases} \mathrm{tansgm}\left(\overline{y}_{i,k}\right) & \text{If the } i\text{th output unit is tansgm} \\ \overline{y}_{i,k} & \text{If the ith output unit is linear} \end{cases}$$

$$\varepsilon_{i,k}^o \leftarrow \begin{cases} \left(y_{i,k}-\hat{y}_{i,k}\right)\left(1-\hat{y}_{i,k}^{\ 2}\right) & \text{If the ith output unit is tansgm} \\ y_{i,k}-\hat{y}_{i,k} & \text{If the ith output unit is linear} \end{cases}$$

For $j=1,2,\cdots,n_h$ and $m=1,2,\cdots,n_o$:

$$\left\langle \tilde{h}_{j,C_m} \right\rangle \leftarrow \frac{1}{|C_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m} \tilde{h}_{j,k}$$

For $i=1,2,\cdots,n_i$:

$$\left\langle o_{ji,C_m} \right\rangle \leftarrow \frac{1}{|C_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m} o_{ji,k}$$

For $j=1,2,\cdots,n_h$ and $i=1,2,\cdots,n_i$:

$$\upsilon_{ji} \leftarrow \upsilon_{ji} - \beta_E\alpha\beta_h\sum_{m=1}^{n_o}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m}\left(\left\langle \tilde{h}_{j,C_m} \right\rangle - \tilde{h}_{j,k}\right)\left(\left\langle o_{ji,C_m} \right\rangle - o_{ji,k}\right)$$

---- updating the weight υ_{ji} according to G

4. Update the quantum intervals:

For $k=1,2,\cdots,K$:

For $j=1,2,\cdots,n_h$:

$$h_{j,k} \leftarrow \sum_{i=1}^{n_i}\upsilon_{ji}x_{i,k}$$

For $r=1,2,\cdots n_s$:

$$h_{j,k}^r \leftarrow \text{tansgm}\left(\beta_h\left(\bar{h}_{j,k} - \theta_j^r\right)\right)$$

$$\tilde{h}_{j,k} \leftarrow \frac{1}{n_s}\sum_{r=1}^{n_s} h_{j,k}^r$$

For $i=1,2,\cdots,n_o$:

$$y_{i,k} \leftarrow \sum_{j=1}^{n_h}\omega_{ij}\tilde{h}_{j,k}$$

$$\hat{y}_{i,k} \leftarrow \begin{cases} \text{tansgm}\left(\bar{y}_{i,k}\right) & \text{If the } i\text{th output unit is tansgm} \\ \bar{y}_{i,k} & \text{If the } i\text{th output unit is linear} \end{cases}$$

$$\varepsilon_{i,k}^o \leftarrow \begin{cases} \left(y_{i,k} - \hat{y}_{i,k}\right)\left(1 - \hat{y}_{i,k}{}^2\right) & \text{If the } i\text{th output unit is tansgm} \\ y_{i,k} - \hat{y}_{i,k} & \text{If the } i\text{th output unit is linear} \end{cases}$$

For $r=1,2,\cdots,n_s$:

$$\kappa_{j,k}^r \leftarrow h_{j,k}^r{}^2 - 1$$

$$\theta_j^r \leftarrow \theta_j^r + \beta_G\frac{\alpha_\theta\beta_h}{n_s}\kappa_{j,k}^r\sum_{i=1}^{n_o}\varepsilon_{i,k}^o\omega_{ij}$$

---- updating the quantum interval according to E

For $j=1,2,\cdots,n_h$ and $m=1,2,\cdots,n_o$:

$$\left\langle \tilde{h}_{j,C_m} \right\rangle \leftarrow \frac{1}{|C_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m} \tilde{h}_{j,k}$$

For $r=1,2,\cdots,n_s$:

$$\left\langle \kappa_{j,C_m}^r \right\rangle \leftarrow \frac{1}{|C_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m} \kappa_{j,k}^r$$

For $j=1,2,\cdots,n_h$ and $r=1,2,\cdots,n_s$:

$$\theta_j^r \leftarrow \theta_j^r - \frac{\alpha_\theta\beta_h}{n_s}\sum_{m=1}^{n_o}\sum_{\mathbf{x}_k:\mathbf{x}_k\in C_m}\left(\left\langle \tilde{h}_{j,C_m} \right\rangle - \tilde{h}_{j,k}\right)\left(\left\langle \kappa_{j,C_m}^r \right\rangle - \kappa_{j,k}^r\right)$$

---- updating the quantum interval according to G.