Quantum Gate Circuit Neural Network Optimization Algorithm Based on Performance Function

Xuan HOU¹,²,* , Shao-song WAN¹,² and Rui LIU¹,³

¹College of Science, Xijing University, Xi’an 710123, China
²Research and Development Center of Intelligent Control Technology, School of Science, Xijing University, Xi’an, 710123, China
³Shaanxi Engineering Research Center of Controllable Neutron Source, School of Science, Xijing University, Xi’an, 710123, China

*Corresponding author

Keywords: Quantum Neural Network, Error Function, Performance Function, Quantum Gate, Pattern Recognition.

Abstract. The current status of Quantum Neural Network (QNN) research is analyzed, it deeply studies the model Quantum Gate Circuit Neural Network (QGCNN) and QGCNN Learning Algorithm (QGCA). Replacing the mean square error function by using the Widrow function and the Rumelhart function, it creates two quantum neural network learning algorithms based on performance function, including Learning Algorithm of QGCNN based on the Widrow functions (WQGCA) and Learning Algorithm of QGCNN based on the Rumelhart functions (RQGCA). New algorithms overcome the inherent defects of the training result is not ideal and unrealistic, which is because the use of mean square error function training network will appear excessive punishment phenomenon. Three algorithms are trained by three data sets. Simulation Experiments on three kinds of algorithms was carried out by using these data sets in the case of the best learning rate. Research prove WQGCA and RQGCA have better pattern classification ability relative to QGCA and RQGCA has a higher classification accuracy than WQGCA.

Introduction

Quantum neural network is a new way to calculation of artificial neural network and quantum computing. Quantum computation is considered as one of the effective ways to improve neural computation [¹-³]. In the strict sense, quantum neural network is a neural network completely constructed by quantum computing mechanism. Because quantum computing can’t be realized by ordinary computer, the neural network constructed by the mechanism can’t be simulated at present [⁴-⁶]. In the usual sense, the quantum neural network refers to a neural network model that introduces the quantum computing mindset into the classical method and is designed to run on a common computer, also known as Quantum-inspired Neural Network [⁷].

Quantum Gate Circuit Neural Network (QGCNN) uses the quantum theory directly to design the neural network topology and training algorithm, it does not have neuron connection weights in classical neural network or quantum neural network based on classical neuron, the classic neural network connection matrix in QGCNN considered as two quantum gate matrices approximately, and the essence of weight updating is to update the corresponding angle parameters in the matrix [², ⁸]. The error function should be a simple addition function. Theoretically, the mean square error function can be used to train the neural network. However, due to the inherent characteristic of the mean square error function, the phenomenon of excessive punishment will occur in the network training process, so the training result is not ideal, even unrealistic.

It proposes two methods to train the network by replacing the mean square error function with two kinds of performance functions: Widrow and Rumelhart function. It establishes two optimization algorithm, including based on Widrow performance function (WQGCA) and based on Rumelhart performance function (RQGCA). The two algorithms overcome the inherent shortcomings of using the mean square error function to train the network.
The mean square error function is defined as

\[ \text{MSEQ}_{\text{GCN}} = \frac{1}{2} \sum_{v=1}^{m} \left( T^{(i)} \cdot A_{iv}^{(1)} \right) \]

Gradient descent method to calculate the angle gradient can be described as

\[ \nabla \text{MSEQ}_{\text{GCN}}(\omega) = \frac{\partial \text{MSEQ}_{\text{GCN}}}{\partial \omega_{g}} = \sum_{v=1}^{m} \left( T^{(i)} \cdot A_{iv}^{(1)} \right) A_{iv}^{(1)} \cot \left( v_{g} + \gamma_{g} \right) A_{iv}^{(1)} \cot \left( 1 - A_{iv}^{(1)} \right) \]

\[ \nabla \text{MSEQ}_{\text{GCN}}(\gamma) = \frac{\partial \text{MSEQ}_{\text{GCN}}}{\partial \gamma_{g}} = \left( T^{(i)} \cdot A_{iv}^{(1)} \right) A_{iv}^{(1)} \cot \left( v_{g} + \gamma_{g} \right) \]

When \( \alpha \) is learning rate, the angle is updated as

\[ \omega_{gj}(k+1) = \omega_{gj}(k) - \alpha \nabla \text{MSEQ}_{\text{GCN}}(\omega) , \quad \gamma_{g}(k+1) = \gamma_{g}(k) - \alpha \nabla \text{MSEQ}_{\text{GCN}}(\gamma) \]

**Optimization Algorithm of QGCNN Based on Performance Function**

In general, it is not ideal or even impractical to adopt the method of finding the mean square error in network training, so we need to replace the mean square error function with some kind of estimation function. The Widrow function and the Rumelhart function are two typical performance functions. For the Widrow function, it estimates the mean square error by using the square of the response error of the neural network under a single learning mode. For the Rumelhart function,
when the knowledge set, that is the total learning mode, is given, the average of the sum of squares of output errors for all learning modes \(^9\).

**Widrow Performance Function.** The Widrow function \(^9\) takes the square of the neural network response error under single learning mode excitation as an estimate of the mean square error. Assuming that the number of QGCNN learning iterations is \(k=1,2,\ldots\), the Widrow performance function is described as

\[
\text{MSE}_{\text{Widrow}} = e^T(k)e(k) = \left(T-A^{(1)}(k)\right)^T\left(T-A^{(1)}(k)\right) = \sum_{v=1}^{m} \left(T_v-\text{A}^{(1)}_v(k)\right)^2 = \sum_{v=1}^{m} e_v^2(k) \quad (5)
\]

When using the Widrow function to build the performance function \(\text{MSE}_{\text{Widrow}}\) as an estimation function of the mean square error function MSE, the network can only learn one sample of all training samples at a time.

**Rumelhart Performance Function.** The Rumelhart performance function is described as

\[
\text{MSE}_{\text{Rumelhart}} = \frac{1}{s}\sum_{i=1}^{s} e^T(k,i)e(k,i) = \frac{1}{s}\sum_{i=1}^{s} \left(T^{(i)}/\text{A}^{(1)}_i(k)\right)^T\left(T^{(i)}/\text{A}^{(1)}_i(k)\right) = \frac{1}{s}\sum_{i=1}^{s} \sum_{v=1}^{m} e_v^2(k,i) \quad (6)
\]

Using the Rumelhart function \(^9\) to build the performance function \(\text{MSE}_{\text{Rumelhart}}\) as an estimation function of the mean square error function MSE, the network remembers all training samples every iteration of learning.

**Optimization Algorithm of QGCNN based on Widrow Performance Function.** QGCNN using the square error function \(\text{MSE}_{\text{QGC}}\), according to the gradient descent principle, the network angle parameters \(\omega\) and \(\gamma\) gradient descent algorithm described as

\[
\begin{align*}
\Delta \omega(k) &= -\alpha \nabla \text{MSE}(\omega(k)) \\
\Delta \gamma(k) &= -\alpha \nabla \text{MSE}(\gamma(k))
\end{align*}
\]

\(\nabla \text{MSE}(\omega(k))\) and \(\nabla \text{MSE}(\gamma(k))\) are the gradient of the mean square error function \(\text{MSE}_{\text{QGC}}\) at \(\omega(k)\) and \(\gamma(k)\). Therefore, the gradient descent learning algorithm of QGCNN is described as

\[
\begin{align*}
\Delta \omega_{jg}(k) &= -\alpha \nabla \text{MSE}(\omega_{jg}(k)) \\
\Delta \gamma_{vg}(k) &= -\alpha \nabla \text{MSE}(\gamma_{vg}(k))
\end{align*}
\]

Quantum state output and real output of QGCNN can be described as

\[
\begin{align*}
\mathbf{\hat{A}}_{iv}^{(1)}(k) &= |0\rangle \cos \theta_v + |1\rangle \prod_{g=1}^{q} \sin \left(\nu_v + \gamma_{vg}\right) = |0\rangle \cos \theta_v + |1\rangle \sin \theta_v \\
\mathbf{\hat{A}}_{iv}^{(1)}(k) &= \mathbf{U} \mathbf{\hat{A}}_{iv}^{(1)}(k) = \sin \theta_v = \prod_{g=1}^{q} \sin \left(\nu_v + \gamma_{vg}\right) = \prod_{g=1}^{q} \sin \left(\arcsin \left(\prod_{j=1}^{n} \sin \left(\theta_{ij} + \omega_{jg}\right)\right) + \gamma_{vg}\right)
\end{align*}
\]

\(\nu_v = \arcsin \left(\prod_{g=1}^{q} \sin \left(\theta_v + \omega_{jg}\right)\right)\), \(\theta_v = \arcsin \left(\prod_{g=1}^{q} \sin \left(\nu_v + \gamma_{vg}\right)\right)\), \(\mathbf{U} = \left(\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right)\)

According to formula (8) and formula (5), the Widrow function of QGCNN is described as

\[
\begin{align*}
\text{MSE}_{\text{Widrow}} &= \sum_{v=1}^{m} \left(T_v^{(i)} - \text{A}^{(1)}_v(k)\right)^2 = \sum_{v=1}^{m} \left(T_v^{(i)} - \prod_{g=1}^{q} \sin \left(\arcsin \left(\prod_{j=1}^{n} \sin \left(\theta_{ij} + \omega_{jg}\right)\right) + \gamma_{vg}\right)\right)^2 \\
\text{The angular gradient is described as}
\end{align*}
\]

\[
\nabla \text{MSE}_{\text{Widrow}}(\omega_{jg}(k)) = \frac{\partial \text{MSE}_{\text{Widrow}}}{\partial \omega_{jg}(k)}, \quad \nabla \text{MSE}_{\text{Widrow}}(\gamma_{vg}(k)) = \frac{\partial \text{MSE}_{\text{Widrow}}}{\partial \gamma_{vg}(k)}
\]

**Steps of WQGC** are as follows

**Step 1** Initialize the condition settings

**Step 1.1** Quantum state description in real sample space
Step 1.2 Set the initial values of the hidden layer matrix \( U^{(0)}(\omega) \) and output layer matrix \( U^{(1)}(\gamma) \)

Step 1.3 Set initial value of network training iteration \( k = 1 \), learning rate \( \alpha \) and defined error \( \varepsilon \)

Step 2 Calculate the quantum state output and real value output of the hidden layer and the output layer of the network

Step 3 Calculate the performance function error \( \text{MSE}^{\text{Widrow}} \) and angle gradient \( \nabla \text{MSE}^{\text{Widrow}} \)

Step 4 Angle updating

\[
\text{MSE}^{\text{Widrow}} = \sum_{q=1}^{m} \left( T^{(i)}_v - \prod_{g=1}^{q} \sin \left( \arcsin \left( \prod_{j=1}^{q} \sin \left( \theta_{ij}^{(i)} + \omega_{jg} \right) \right) + \gamma_{vg} \right) \right)^2
\]

\[
\begin{align*}
\nabla \text{MSE}^{\text{Widrow}} (\omega_{jg}(k)) &= -\frac{\partial \text{MSE}^{\text{Widrow}} (\omega_{jg}(k))}{\partial \omega_{jg}(k)} \\
\Delta \omega_{jg}(k) &= -\alpha \nabla \text{MSE} (\omega_{jg}(k)) \\
\omega_{jg}(k+1) &= \omega_{jg}(k) + \Delta \omega_{jg}(k)
\end{align*}
\]

\[
\begin{align*}
\nabla \text{MSE}^{\text{Widrow}} (\gamma_{vg}(k)) &= -\frac{\partial \text{MSE}^{\text{Widrow}} (\gamma_{vg}(k))}{\partial \gamma_{vg}(k)} \\
\Delta \gamma_{vg}(k) &= -\alpha \nabla \text{MSE} (\gamma_{vg}(k)) \\
\gamma_{vg}(k+1) &= \gamma_{vg}(k) + \Delta \gamma_{vg}(k)
\end{align*}
\]

Step 5 Unconditional transfer: \( k = k+1 \), go to step 2

Step 6 Shutdown test: If \( \text{MSE}^{\text{Widrow}} \leq \varepsilon \), then stop training

**Optimization Algorithm of QGCNN based on Rumelhart Performance Function.** Referring to the derivation of WQGCA, according to formula (8) and formula (6), the Rumelhart function of QGCNN is described as

\[
\text{MSE}^{\text{Rumelhart}} = \frac{1}{s} \sum_{i=1}^{s} \sum_{g=1}^{m} \left( T^{(i)}_v - \prod_{j=1}^{g} \sin \left( \arcsin \left( \prod_{j=1}^{g} \sin \left( \theta_{ij}^{(i)} + \omega_{jg} \right) \right) + \gamma_{vg} \right) \right)^2
\]

(11)

The angular gradient is described as

\[
\nabla \text{MSE}^{\text{Rumelhart}} (\omega_{jg}(k)) = -\frac{\partial \text{MSE}^{\text{Rumelhart}} (\omega_{jg}(k))}{\partial \omega_{jg}(k)} , \quad \nabla \text{MSE}^{\text{Rumelhart}} (\gamma_{vg}(k)) = -\frac{\partial \text{MSE}^{\text{Rumelhart}} (\gamma_{vg}(k))}{\partial \gamma_{vg}(k)}
\]

(12)

Steps of RQGCA are as follows

Step 1~Step 2: Refer to Step 1 and Step 2 of WQGCA

Step 3 Calculate the performance function error \( \text{MSE}^{\text{Rumelhart}} \) and angle gradient \( \nabla \text{MSE}^{\text{Rumelhart}} \)

Step 4 Angle updating

\[
\text{MSE}^{\text{Rumelhart}} = \frac{1}{s} \sum_{i=1}^{s} \sum_{g=1}^{m} \left( T^{(i)}_v - \prod_{j=1}^{g} \sin \left( \arcsin \left( \prod_{j=1}^{g} \sin \left( \theta_{ij}^{(i)} + \omega_{jg} \right) \right) + \gamma_{vg} \right) \right)^2
\]

\[
\begin{align*}
\nabla \text{MSE}^{\text{Rumelhart}} (\omega_{jg}(k)) &= -\frac{\partial \text{MSE}^{\text{Rumelhart}} (\omega_{jg}(k))}{\partial \omega_{jg}(k)} \\
\Delta \omega_{jg}(k) &= -\alpha \nabla \text{MSE} (\omega_{jg}(k)) \\
\omega_{jg}(k+1) &= \omega_{jg}(k) + \Delta \omega_{jg}(k)
\end{align*}
\]

\[
\begin{align*}
\nabla \text{MSE}^{\text{Rumelhart}} (\gamma_{vg}(k)) &= -\frac{\partial \text{MSE}^{\text{Rumelhart}} (\gamma_{vg}(k))}{\partial \gamma_{vg}(k)} \\
\Delta \gamma_{vg}(k) &= -\alpha \nabla \text{MSE} (\gamma_{vg}(k)) \\
\gamma_{vg}(k+1) &= \gamma_{vg}(k) + \Delta \gamma_{vg}(k)
\end{align*}
\]

Step 5 Unconditional transfer: \( k = k+1 \), go to step 2

Step 6 Shutdown test: If \( \text{MSE}^{\text{Rumelhart}} \leq \varepsilon \), then stop training

**Simulation Experiment and Analysis**

**Experimental Model.** Using Iris, Wine and CarEvaluation data sets to analyze the pattern recognition ability of QGCA, WQGCA and RQGCA, the parameters are set as shown in Table 1.
Table 1. Parameters of QGCNN

| Parameters          | Iris | Wine | CarEvaluation | Parameters          | Iris | Wine | CarEvaluation |
|---------------------|------|------|---------------|---------------------|------|------|---------------|
| Input layer neurons | 2    | 7    | 3             | Limit the error ε   | 0.015| 0.012| 0.005         |
| Hidden layer neurons| 2×8  | 7×13 | 3×6           | Learning rate α     | 0.05, 0.1, 0.15, 0.2, 0.25, …, 1.0 |
| Output layer neurons| 3    | 3    | 4             |                     |      |      |               |

**Result Analysis.** As shown in Figure 2 to Figure 4, the best learning rates of QGCNN are 0.6 (QGCA), 0.5 (WQGCA) and 0.7 (RQGCA). The research proves that, for QGCNN, RQGCA and WQGCA are better than QGCA.

![Figure 2. α influence on k with Iris data set](image1)

Figure 2. α influence on k with Iris data set

![Figure 3. α influence on k with Wine data set](image2)

Figure 3. α influence on k with Wine data set

![Figure 4. α influence on k with CarEvaluation data set](image3)

Figure 4. α influence on k with CarEvaluation data set

**Classification Test.** All of best learning rates are used for the simulation test. The results show that classification accuracy of RQGCA and WQGCA are better than QGCA, and the accuracy of RQGCA is higher relative to WQGCA, as shown in Table 2.

Table 2. Classification test results

| Data set       | Iris | Wine | CarEvaluation |
|----------------|------|------|---------------|
| Learning algorithm | QGCA | WQGCA | RQGCA | QGCA | WQGCA | RQGCA | QGCA | WQGCA | RQGCA |
| Test sample    | 75   | 75   | 75           | 89   | 89   | 89   | 864  | 864  | 864   |
| Correct classification | 73   | 74   | 74           | 83   | 86   | 87   | 845  | 855  | 861   |
| Correct rate   | 97.3%| 98.7%| 98.7%        | 93.3%| 96.6%| 97.8%| 97.8%| 98.9%| 99.7% |

**Conclusion**

Through the further study of QGCNN and its basic learning algorithm (QGCA), it proposes to replace the mean square error function with two kinds of performance functions: Widrow function and Rumelhart function, and establish two kinds of quantum gate lines based on performance function neural network learning algorithms, namely WQGCA and RQGCA. The two algorithms overcome the phenomenon of over-punishing training networks by using the mean Square Error Function, so that there is an inherent defect that the training result is not ideal and unrealistic. Three algorithms were trained by using Iris, Wine and CarEvaluation datasets. Under the condition of obtaining the best learning rate, the three algorithms were classified and simulated by using datasets.
The results show that WQGCA and RQGCA have better model classification ability than QGCA, and RQGCA has higher classification accuracy than WQGCA.

Acknowledgement

This research was financially supported by the National Natural Science Foundation of China under Grant No. 51507186.

References

[1] JIAO Licheng, YANG Shuyuan, LIU Fang, et al. Seventy Years Beyond Neural Networks: Retrospect and Prospect [J]. Chinese Journal of Computers, 2016,39(8):1697-1716.

[2] LI Shiyong, LI Panchi. Quantum Computation and Quantum Optimization Algorithms [M]. Harbin: Harbin Institute of Technology Press, 2009

[3] LI Panchi, ZHOU Hongyan. Model and Algorithm of Quantum Neural Network Based on the Controlled Hadamard Gates [J]. Journal of Computer Research and Development, 2015,52(1):211-220.

[4] LI Nan, HOU Xuan. Research on Adaptive Quantum Forward Counter Propagation Algorithm [J]. Journal of Electronics & Information Technology, 2013,35(11):2778-2783.

[5] LI Sheng, ZHANG Peilin, LI Bing, et al. Quantum neural network model based on quantum gate set and its application. Computer Engineering and Applications, 2015, 51(6): 33-36.

[6] Zhang Yipeng, Chen Liang, Hao Huan. An Improved Training Algorithm for Quantum Neural Networks [J]. Journal of Electronics & Information Technology, 2013,35(7):1630-1635.

[7] LI Panchi, LI Guorui. Hybrid Quantum-inspired Neural Networks Model and Algorithm [J]. Journal of Electronics & Information Technology, 2016,38(1):111-118.

[8] HOU Xuan. Research on quantum gate circuit neural network and improved learning algorithm [J]. Computer Engineering and Applications, 2014, 50(6):213-218.

[9] RUAN Xiaogang. Neural Computing Science [M]. National Defense Industry Press, 2006.

[10] LI Penghua, CHAI Yi, XIONG Qingyu. Quantum Gate Elman Neural Network and Its Quantized Extended Gradient Back-propagation Training Algorithm [J]. ACTA Automatica Sinica, 2013,39(9):1511-1522.

[11] LI Panchi, SONG Kaoping, YANG Erlong. Quantum neural networks model and algorithm based on quantum gates circuit [J]. Control and Decision, 2012,27(1):143-146,151.

[12] ZHANG Peilin, LI Sheng, WU Dinghai. Net model of restricted boltzmann machine based on quantum computation and its classification method [J]. Journal of Vibration and Shock, 2015,34(24):26-31.

[13] ZHANG Liang, LU Yuliang, FANG Shanyao. Research on Deep Web Classification Approach Based on Quantum Self-organization Feature Mapping Network [J]. Computer Science, 2011,38(6):205-210.

[14] LI Panchi, SHI Guangyao. Sequence-Input Based Quantum Neural Networks Model and Its Algorithm [J]. Pattern Recognition & Artificial Intelligence, 2013,26(3):247-253.

[15] ZHANG Lifang, ZHANG Xiping. Network Traffic Prediction Based on BP Neural Networks Optimized by Quantum Genetic Algorithm [J]. Computer Engineering & Science, 2016,38(1):114-119.