# Hybrid Quantum Neural Networks Model Algorithm and Simulation

Hong Xiao
School of Computer & Information Technology
Daqing Petroleum Institute
Daqing, Heilongjiang, China
Email: xh_daqing@126.com

Maojun Cao
School of Computer & Information Technology
Daqing Petroleum Institute
Daqing, Heilongjiang, China
Email: caomaojun@126.com

*Abstract*—**A quantum neural networks model based on quantum neurons and traditional neurons is presented in this paper. The input of quantum neuron is real vector, its weight is quantum bits, its transforming function is an inner product operator and its output is a real number. The network includes three layers. Input layer is composed of traditional neurons that receive input information. Hidden layer is composed of quantum neurons that extract pattern feature of input information and transfer them to output layer. Output layer is composed of traditional neurons that export calculation result. The weightings of output layer are rectified by back propagation algorithm. The weightings of hidden layer are rectified by a group of quantum gates. A detailed learning algorithm is designed. Finally the availability of the model and algorithm is illustrated by two application examples of pattern recognition and functional approximation.**

***Keywords- quantum neuron; quantum neural networks; quantum computing; quantum gates***

## I. INTRODUCTION

In the eighties of the twentieth century, Benioff and Feynman proposed the concept of quantum computation, then P.W. Shor gave the first quantum algorithm of very large integer factorization in 1994 and L.K. Grover proposed a quantum algorithm which searches a marked state in an unordered list in 1996, quantum computation has been widely paid attention and become a challenging research line. Fuzzy logic, evolution calculation, and neural network are regard as the most promising three important aspects in the artificial intelligence field, which compose intelligence calculation (soft-calculation) and exists much comparability with quantum calculation. Therefore, the amalgamation of them would bring promising research line in theory. At present, elementary amalgamation of quantum calculation and evolution calculation has already existed and gained some valuable research results [1-3], however, as yet, there is little understanding of the essential components of artificial neural networks (ANNs) based on quantum theoretical concepts and techniques. Quantum perceptions were proposed by Lewestein [4], where instead of classical weights in the perception a unitary operator is used to map inputs to outputs. During training the unitary operator is developed to find the correct mapping. The use of quantum theoretical techniques in ANNs for cognitive modeling has also been suggested [5] but these are predominantly theoretical proposals arguing the need for quantum neural computing. Quantum learning was proposed by Chrisley [6]. While he outlines practical concerns for the implementation of such a process, he does not test the success of quantum learning by simulation. Menneer and Narayanan [7,8] have extended Chrisley's design in their proposal for single pattern quantum neural networks. Recently, Ezhov [9] argues that current approaches to quantum neural networks based on such single pattern networks are consistent with the Everett parallel universe interpretation of quantum mechanics. Overall, it therefore appears that, apart from Menneer and Narayanan [7,8], no experimentation or simulation has yet been done on quantum neural networks to determine their architecture and function or even their power over their classical counterparts [10].

The research presented in this paper (a) constructs quantum neuron and quantum neural network model corresponding to traditional forward-feed network model, (b) proposes an algorithm for this model, and (c) designs two experiments, the results demonstrate the model proposed in this paper is efficient.

## II. HYBRID QUANTUM NEURAL NETWORKS MODEL

### *A. Quantum neuron model*

A neuron can be described as a four elements array: (input, weight, transform function, output), where input and output is the outer attribute of the neuron, and weight and transform function are the inner attribute of the neuron. Therefore, the different neuron models can be constructed by modifying types of weight and transform function. According this viewpoint, for the quantum neuron proposed in this paper, the weight is represented by qubits, and the transform function is represented by inner-product operator. At the same time, the difference from the traditional neuron is that the quantum neuron carries a group of one-qubit gates that modify the phase of weight qubits. The quantum neuron model is presented in Fig.1.

In quantum neuron model, the weight is represented by qubit $|\phi_i\rangle$, a qubit can be described as follow:

$$|\phi_i\rangle = \alpha_i|0\rangle + \beta_i|1\rangle,\ (i = 1,2,\cdots,n) \tag{1}$$

where $\alpha_i$ and $\beta_i$ are complex numbers, $|\alpha_i|^2$ and $|\beta_i|^2$ give the probabilities that the qubit will respectively be found in the $|0\rangle$ and $|1\rangle$, and satisfy the normalization conditions as follow:

$$|\alpha_i|^2 + |\beta_i|^2 = 1, \ (i = 1,2,\cdots,n) \qquad (2)$$

The numbers $\alpha_i$ and $\beta_i$ satisfied Eq.(2) are called the probability amplitudes of the qubit corresponding states. Therefore, the qubit can also be described by the probability amplitudes as $[\alpha_i, \beta_i]^T$.

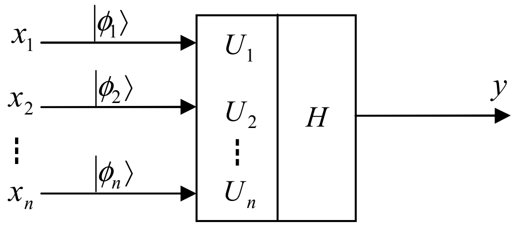

Figure 1. Quantum neuron

**Definition**: Suppose $X = (x_1, x_2, \cdots, x_n)^T$, $x_i$ is real number, $W = (w_1, w_2, \cdots, w_n)^T$, $w_i$ is $m$ dimension real vector, namely, $w_i = (w_{i1}, w_{i2}, \cdots, w_{im})^T$. $XW$ is defined as Eq.(3).

$$XW = \sum_{i=1}^{n} x_i w_i \qquad (3)$$

where $x_i w_i$ is computed according to algorithm between scalar quantity and vector.

Let $X = (x_1, x_2, \cdots, x_n)^T$ is the input real vector, $y$ is a output real number, $W = (w_1, w_2, \cdots, w_n)^T$ is the weight matrix, $w_i$ is probability amplitudes vector of qubit $|\phi_i\rangle$ $(i = 1,2,\cdots,n)$, the in-out relation of quantum neuron can be described as follow:

$$y = H(XW) = C \bullet (XW) = C \bullet \sum_{i=1}^{n} x_i |\phi_i\rangle \qquad (4)$$

where $H$ is an inner product operator; $H(X) = C \bullet X$; $C = (1\ ,1)^T$. The operator, as transform function, transforms the input of quantum neuron to a real number. $U_i$ is a quantum rotation gate to modify the phase of the $|\phi_i\rangle$

### B. *Hybrid quantum neural networks model*

Quantum Neural Network (QNN) is defined as the model that all the input, output, and linked weights for each layer may be qubits. The QNN structure is the same as the general ANN which includes input layer, hidden layer, and output layer. Obvious, the network including neuron in Fig.1 is QNN. The QNN only including quantum neurons is defined as the normalization QNN, and the QNN including quantum neurons and general neurons is defined as the hybrid QNN. Since QNN transform function adopts linear operator, the nonlinear mapping capability falls under restriction. Therefore, this paper considers the hybrid QNN including a hidden layer of quantum neuron, which holds the advantage of the quantum computing and the nonlinear mapping capability of the general ANN. QNN model is presented in Fig.2.

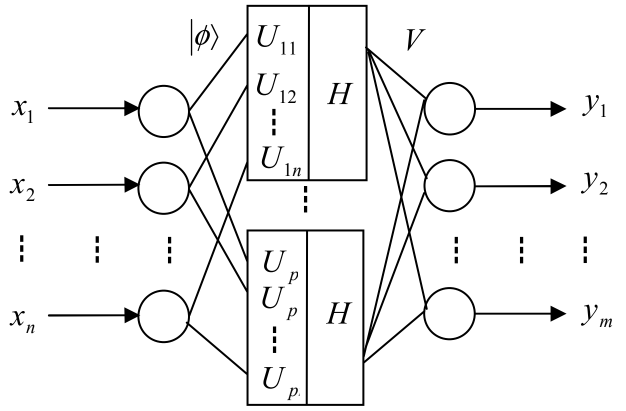

Figure 2. Hybrid Quantum neural networks medel

The model includes three layers, the input layer and output layer contains $n$ and $m$ traditional neurons, respectively, and the hidden layer contain $p$ quantum neurons. The in-out relation of the QNN can be described as follow:

$$y_i = g\Big(\sum_{j=1}^{p} v_{ij}\big(C \bullet \sum_{k=1}^{n} (x_k |\phi_{jk}\rangle)\big)\Big) \qquad (5)$$

where $i = 1,2,\cdots,m$ ; $j = 1,2,\cdots,p$ ; $k = 1,2,\cdots,n$ , $v_{ij}$ is the linked weight between the $i$ th neuron in output layer and the $j$ th neuron in hidden layer, $g$ is Sigmoid function or Gauss function.

## III. HYBRID QUANTUM NEURAL NETWORKS LEARNING ALGORITHM

For the QNN in Fig.2, when transform function in output layer is continuous and differentiable, the learning algorithm may adopt BP algorithm. The output error function is defined as follow:

$$E = \frac{1}{2} \sum_{k=1}^{m} (\hat{y}_k - y_k)^2 \qquad (6)$$

where $\hat{y}_k$ denotes anticipant output.

According to BP algorithm, the modifying formula for output layer weight is described as follow:

$$v_{jk}(t+1) = v_{jk}(t) + \eta_1 \frac{\partial E}{\partial v_{jk}}$$
$$= v_{jk}(t) + \eta_1(\hat{y}_k - y_k(t))g'(t)h_j(t) \qquad (7)$$

where, $j = 1,2,\cdots,p$ , $k = 1,2,\cdots,m$ , $\eta_1$ denotes learning ratio, and $t$ denotes learning steps.

The weights between the input layer and quantum hidden layer are modified by quantum gates described as follow:

$$U_{ij} = \begin{bmatrix} \cos(\Delta\theta_{ij}) & -\sin(\Delta\theta_{ij}) \\ \sin(\Delta\theta_{ij}) & \cos(\Delta\theta_{ij}) \end{bmatrix} \qquad (8)$$

where $i = 1,2,\cdots,n$ , $j = 1,2,\cdots,p$ .

According BP algorithm, the $\Delta\theta_{ij}$ can be gained from Eq.(9)

$$\Delta\theta_{ij}(t) = -\frac{\partial E}{\partial \theta_{ij}} = -\sum_{k=1}^{m} \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial h_j}\frac{\partial h_j}{\partial \theta_{ij}}$$
$$= -\sum_{k=1}^{m} (\hat{y}_k - y_k(t))g'(t)v_{jk}(t)h_j(t)x_i$$
$$(\cos(\gamma_{ij}(t)) - \sin(\gamma_{ij}(t))) \qquad (9)$$

where $\gamma_{ij}(t)$ is the phase of the $|\phi_{ij}\rangle$ .

For $\gamma_{ij}(t)$ , the modifying formula is described as follow:

$$\gamma_{ij}(t+1) = \gamma_{ij}(t) + \eta_2\Delta\theta_{ij}(t) \qquad (10)$$

Let the vector of $|\phi_{ij}\rangle$ is $[\alpha_{ij}, \beta_{ij}]^T$ , the modifying formula of $|\phi_{ij}\rangle$ is described as follow:

$$\begin{bmatrix} \alpha_{ij}(t+1) \\ \beta_{ij}(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\eta_2\Delta\theta_{ij}) - \sin(\eta_2\Delta\theta_{ij}) \\ \sin(\eta_2\Delta\theta_{ij}) \ \ \cos(\eta_2\Delta\theta_{ij}) \end{bmatrix} \begin{bmatrix} \alpha_{ij}(t) \\ \beta_{ij}(t) \end{bmatrix} \qquad (11)$$

where $\eta_2$ denotes learning ratio, $t$ denotes learning steps.

Synthesizing above depiction, the QNN learning algorithm can be described as follows:

**Step1**. The network parameters are initialized. Including: the number of nodes in each layer, the learning rate, the restriction error $\varepsilon$ , the restriction iterations $Max$ . Set the current iteration $t = 0$ .

**Step2**. The network weights are initialized.

Hidden layer: $\gamma_{ij}(0) = 2\pi \times Rnd$ ,

$|\phi_{ij}\rangle = [\cos(\gamma_{ij}(0)) \ \sin(\gamma_{ij}(0))]^T$ ,

Output layer: $v_{jk} = Rnd - 0.5$

where $i = 1,2,\cdots,n$ ; $j = 1,2,\cdots,p$ ; $k = 1,2,\cdots,m$ ; $Rnd$ is a random number in (0,1)

**Step3.** Compute network output according to Eq.(5), modify the weights of each layer according to Eq.(7) and Eq.(11), respectively.

**Step4.** Compute network output error according to Eq.(6), if $E < \varepsilon$ or $t > Max$ then go to **Step5,** else $t = t+1$, go to **Step3**

**Step5** Save the weight value, stop.

## IV. COMPARISON EXPERIMENT

To testify the validity of QNN, two kinds of experiments are designed and the QNN is compared with the general BP network in this part. In the experiments, the QNN adopts the same structure and parameters as the BP network.

### A. *Pattern recognition*

**Experiment 1.** Nine-sample patterns classification.

For Nine sample points in Fig.3, determine the pattern of each point by QNN. This is a typical two-pattern classification problem, which is regard as the generalization of XOR problem.

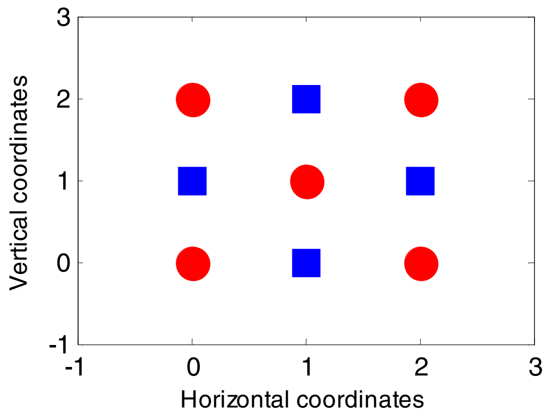

Figure 3. Nine points distribution on the plane

The experimental aim is to determine the pattern of each point. The sample data are shown is Table1.

**Table 1.** The samples data of patterns classification

| Input | 0 0 | 0 1 | 1 0 | 1 1 | 0 2 | 2 0 | 1 2 | 2 1 | 2 2 |
|---|---|---|---|---|---|---|---|---|---|
| Output | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

The network construct is 2-10-1, the learning ratio is 0.8, and the inertia factor is 0.3. The QNN and the BP network is learned 50 times, restively, then compute the average of convergence steps. Learning result is shown in Table2.

**Table 2.** The learning result comparison between QNN and BP network for nine-sample patterns problem (average of 50 times)

| Restriction error | Restriction steps | QNN | | BP | | |
|---|---|---|---|---|---|---|
| | | Convergence steps | Convergence rate | Convergence steps | Convergence rate | Necessary steps for convergence |
| 0.10 | 100 | 65 | 94% | 100 | Non-convergence | 605 |
| 0.05 | 500 | 221 | 98% | 500 | Non-convergence | 1693 |
| 0.01 | 5000 | 3735 | 84% | 5000 | Non-convergence | 14238 |

**Table 3.** The learning result comparison between QNN and BP network for double spiral curve problem (average of 30 times)

| Restriction error | Restriction steps | QNN | | BP | | |
|---|---|---|---|---|---|---|
| | | Convergence steps | Convergence rate | Convergence steps | Convergence rate | Necessary steps for convergence |
| 0.10 | 2000 | 1580 | 80% | 2000 | Non-convergence | 4471 |
| 0.05 | 10000 | 6380 | 90% | 8611 | 73% | — |

**Experiment 2.** Double spiral curves classification problem.

For two spiral curves in Fig.4, fetch 25 points from each curve, respectively, compose the sample set containing 50 sample points, and classify these points in sample set by QNN.

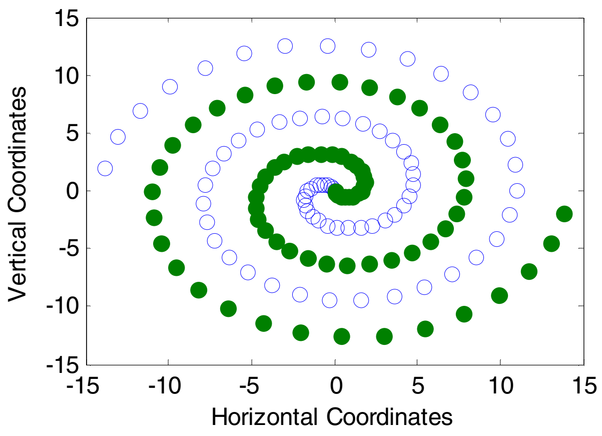

Figure 4. Double spiral curves

The network construct is 2-20-1, the learning ratio is 0.8, and the inertia factor is 0.3. The QNN and the BP network is learned 30 times, restively, then compute the average of convergence steps. Learning result is shown in Table3.

*B. Functional approximation*

Taking example for trigonometric function holding singular value, compute 11 discrete value of the function $f(x)=\cos(1/x)$ in [1/( $2\pi$ +1),1]. These discrete values are shown in Table4.

**Table 3.** 11 discrete values of $f(x)=\cos(1/x)$

| $x$ | $f(x)$ |
|---|---|
| 0.1373 | 0.5403 |
| 0.1503 | 0.9317 |
| 0.1659 | 0.9672 |
| 0.1852 | 0.6333 |
| 0.2096 | 0.0575 |
| 0.2415 | -0.5403 |
| 0.2846 | -0.9317 |
| 0.3466 | -0.9672 |
| 0.4431 | -0.6333 |
| 0.6141 | -0.0575 |
| 1.0000 | 0.5403 |

On the basis of the data in Table4, approximate the function $f(x)=\cos(1/x)$ in [1/( $2\pi$ +1),1] by QNN. The network construct is 2-20-1, the learning ratio is 0.8, the inertia factor is 0.3, the restriction error is 0.05, and the restriction steps are 15000. The QNN reaches convergence after 12242 steps, here, approximation error is 0.0499, and approximation curve is shown in Fig.5. When BP network holding the same structure as the QNN is applied to approximate $f(x)=\cos(1/x)$, the network does not reach to convergence after 15000 steps, and the approximation error is 0.0627.

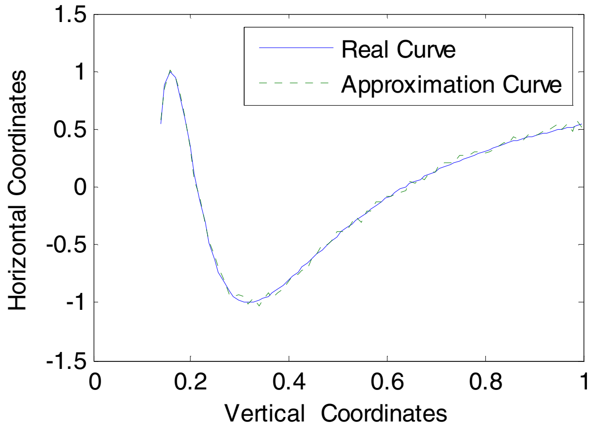

Figure 5. Functional approximation curve

## V. CONCLUSIONS

The QNN is the amalgamation of quantum computing and nerve computing, which holds the advantage such as parallelism and high efficiency of quantum computation besides continuity, approaching capability, and generalization capability that the general ANN holds. In QNN, the weights are represented by qubits, and the phase of each qubit is modified by the quantum rotation gate. Since both probability amplitudes participate in optimizing computation, the computation capability is evidently superior to the general BP network. Experiment proves that the QNN model proposed in this paper is efficient.

## ACKNOWLEDGMENT

The paper was supported by natural science fund project of Heilongjiang Province (Grant No. F2007-11) and Scientific Research Fund of Heilongjiang Provincial Education Department (Grant No. 11521005)

## REFERENCES

[1] A. Narayanan, M. Moore, Quantum inspired genetic algorithms[A]. Proce of the 1996 IEEE International Conference on Evolutionary Computation (ICEC96)[C]. Nogaya: IEEE Press, 1996. 41-46.

[2] K.H. Han, Genetic quantum algorithm and its application to combinatorial optimization problem[A]. IEEE Proc of the 2000 Congress on Evolutionary Computation[C].San Diego: IEEE Press, 2000. 1354-1360.

[3] J.A. Yang, Research of quantum genetic algorithm and its application in blind source separation[J]. ACTA Electronica Sinica (China), 20(1) (2003) 62-68.

[4] M. Lewestein, Quantum perceptrons, Journal of Modern Optics 41 (12) (1994) 2491-2501.

[5] S.C. Kak, Quantum neural computing, Advances in Imaging and Electron Physics 94 (1995)259-314.

[6] R.L. Chrisley, Quantum learning, in: P. Pylkk anen, P. Pylkko (Eds.), New Directions in Cognitive Science, Proceedings of the International Symposium, Saariselk a, Finnish Artificial Intelligence Society, Lapland, Finland, 1995.

[7] T. Menneer, A. Narayanan, Quantum inspired neural networks, Department of Computer Science, University of Exeter, UK, http://www.dcs.ex.ac.uk/reports/reports.html, 1995.

[8] T. Menneer, A. Narayanan, Quantum artificial neural networks vs classical arti®cial neural networks: experiments in simulation, in: Proceedings of the Fifth Joint Conference on Information Sciences, vol. 1, 2000, pp. 757-759.

[9] A. Ezhov, Spurious memory, single-class and quantum neural networks, in: Proceedings of the Fifth Joint Conference on Information Sciences, vol. 1, 2000, pp. 767-770.

[10] A. Narayanan, T. Menneer, Quantum artificial neural network architectures and components, Information Sciences 128 (2000) 231-255.