Quantum-inspired Neural Networks with Applications

Maojun Cao
School of Computer & Information Technology
Northeast Petroleum University
Daqing, Heilongjiang, China

Panchi Li
School of Computer & Information Technology
Northeast Petroleum University
Daqing, Heilongjiang, China
E-mail: lipanchi {at} vip.sina.com

Abstract—**On the basis of analyzing the principles of the quantum rotation gates and quantum controlled-NOT gates, an improved design for CNOT gated quantum neural networks model is proposed and a smart algorithm for it is derived based on the *Levenberg-Marquardt* algorithm in our paper. In improved model, the input information is expressed by the qubits, which, as the control qubits after rotated by the rotation gate, control the qubits in the hidden layer to reverse. The qubits in the hidden layer, as the control qubits after rotated by the rotation gate, control the qubits in the output layer to reverse. The networks output is described by the probability amplitude of state |1〉 in the output layer. It has been shown in two application examples of modeling of acrylamide homogeneous polymerization process and wine recognition that the proposed model is superior to the common BP networks with regard to their convergence ratio, number of iterations, approximation and generalization ability.**

Keywords- quantum computing; quantum neural networks; quantum gate; learning algorithm

I. INTRODUCTION

For over past decades many researches and publications have been dedicated to improve the performance of neural networks. Useful models to enhance the approximation and generalization abilities include: local linear radial basis function neural networks, which replaced the connection weights of conventional radial basis function neural networks by a local linear model [1]; selective neural networks ensemble with negative correlation, which employed the hierarchical pair competition-based parallel genetic algorithm to train the neural networks forming the ensemble [2]; polynomial based radial basis function neural networks [3]; hybrid wavelet neural networks, which employed rough set theory to help in decreasing the computational effort needed for building the networks structure [4]; simultaneous optimization of artificial neural networks, which employed GA to optimize multiple architectural factors and feature transformations of ANN to relieve the limitations of the conventional BP algorithm [5].

Since Kak firstly proposed the concept of quantum neural computation [6] in 1995, the quantum neural networks have attracted great attention during the past decade, and a large number of novel techniques have been studied for the quantum computation and neural networks. For example, Reference [7] proposed the model of quantum neural networks with multi-level hidden neurons based on the superposition of quantum states in the quantum theory. In 1998, a new neural network model with quantum circuit was developed for the quantum computation, and was proven to exhibit a powerful learning capability [8]. Matsui et al. develop a quantum neural networks model using the single bit rotation gate and two-bit controlled-NOT gate. They also investigate its performance in solving the four-bit parity check and the function approximation problems [9]. Reference [10] proposes the neural networks with the quantum gated nodes, and indicates that such quantum networks may contain more advantageous features from the biological systems than the regular electronic devices. In our previous work [11], we have proposed a quantum BP neural networks model with learning algorithm based on the single-qubit rotation gates and two-qubit controlled-NOT gates.

In this paper, we study a new neural networks model with the quantum gated nodes. Our scheme is a three-layer model with a hidden layer, which employs the *Levenberg-Marquardt* (L-M) algorithm for learning. The input-output relationship of this model is derived based on the physical meaning of the quantum gates. The convergence ratio, number of iterations, and approximation error of the quantum neural networks are examined with different learning coefficients. Two application examples demonstrate that this quantum neural network is superior to the common BP neural networks (BPNN).

II. QUANTUM BITS AND QUANTUM GATES

A. Quantum Bits

What is a qubit? Just as a classical bit has a state-either 0 or 1- a qubit also has a state. Two possible states for a qubit are the state $|0\rangle$ and $|1\rangle$, which as you might guess correspond to the states 0 and 1 for a classical bit.

Notation like $|\rangle$ is called the *Dirac* notation, and we will see it often in the following paragraphs, as it is the standard notation for states in quantum mechanics. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. It is also possible to form linear combinations of states, often called superposition

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle , \quad (1)$$

This work was supported by the National Natural Science Foundation of China (Grant No. 61170132).

where $0 \le \theta \le \pi$, $0 \le \phi \le 2\pi$.

Therefore, unlike the classical bit, which can only be set equal to 0 or 1, the qubit resides in a vector space parametrized by the continuous variables θ and ϕ. Thus, a continum of states is allowed. The Bloch sphere representation is useful in thinking about qubits since it provides a geometric picture of the qubit and of the transformations that one can operate on the state of a qubit. Owing to the normalization condition, the qubit's state can be represented by a point on a sphere of unit radius, called the Bloch Sphere. This sphere can be embedded in a three-dimensional space of Cartesian coordinates ($x = \cos\phi\sin\theta$, $y = \sin\phi\sin\theta$, $z = \cos\theta$). By definition, a Bloch vector is a vector whose components (*x*, *y*, *z*) single out a point on the Bloch sphere. We can say that the angles θ and ϕ define a Bloch vector, as shown in Fig.1, where the points corresponding to the following states are shown: $|A\rangle = [1, 0]^{\mathrm{T}}$, $|B\rangle = [0, 1]^{\mathrm{T}}$, $|C\rangle = |E\rangle = [\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]^{\mathrm{T}}$, $|D\rangle = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^{\mathrm{T}}$, $|F\rangle = [\frac{1}{\sqrt{2}}, \frac{-\mathrm{i}}{\sqrt{2}}]^{\mathrm{T}}$, $|G\rangle = [\frac{1}{\sqrt{2}}, \frac{\mathrm{i}}{\sqrt{2}}]^{\mathrm{T}}$.

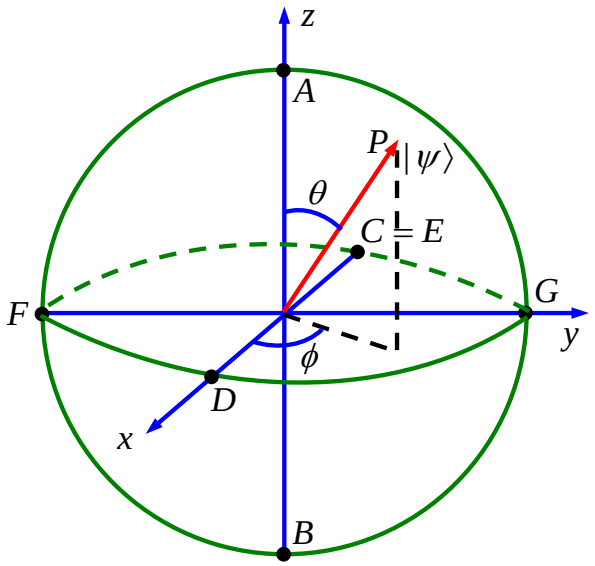

Figure 1. A qubit description on Bloch sphere.

For convenience, in this paper, we represent the qubit's state by a point on a circle of unit radius as shown in Fig.2. The relations between Fig.1 and Fig.2 can be written as

$$\begin{cases} \alpha: 0 \to \pi/2 & \Leftrightarrow \phi = 0 \ and \ \theta: \pi/2 \to 0 \\ \alpha: \pi/2 \to \pi & \Leftrightarrow \phi = \pi \ and \ \theta: 0 \to \pi/2 \\ \alpha: \pi \to 3\pi/2 & \Leftrightarrow \phi = \pi \ and \ \theta: \pi/2 \to \pi \\ \alpha: 3\pi/2 \to 2\pi & \Leftrightarrow \phi = 0 \ and \ \theta: \pi \to \pi/2 \end{cases}. \quad (2)$$

At this time, any state of the qubit may be written as

$$|\varphi\rangle = \cos\alpha\,|0\rangle + \sin\alpha\,|1\rangle\,. \quad (3)$$

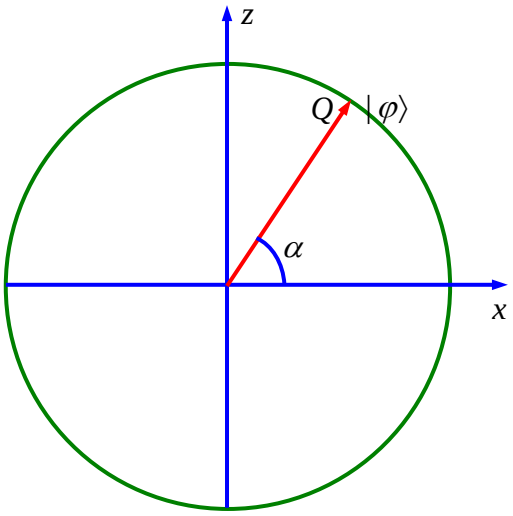

Figure 2. A qubit description on unit circle.

An n-qubits system has 2^n computational basis states. For example, a 2 qubits system has basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. Similar to the case of a single qubit, the n-qubits system may form the superpositions of 2^n basis states

$$|\phi\rangle = \sum_{x \in \{0,1\}^n} a_x\,|x\rangle\,, \quad (4)$$

where a_x is called probability amplitude of the basis states $|x\rangle$, and $\{0,1\}^n$ means the set of strings of length two with each letter being either zero or one.

The condition that these probabilities can sum to one is expressed by the normalization condition

$$\sum_{x \in \{0,1\}^n} |a_x|^2 = 1\,. \quad (5)$$

B. Quantum Rotation Gate

In the quantum computation, the logic function can be realized by applying a series of unitary transform to the qubit states, which the effect of the unitary transform is equal to that of the logic gate. Therefore, the quantum services with the logic transformations in a certain interval are called the quantum gates, which are the basis of performing the quantum computation. A single qubit rotation gate can be defined as

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \quad (6)$$

Let the quantum state $|\phi\rangle = [\cos\theta_0 \ \sin\theta_0]^{\mathrm{T}}$, and $|\phi\rangle$ can be transformed by $R(\theta)$ as follows:

$$R(\theta)|\phi\rangle = \begin{bmatrix} \cos(\theta_0+\theta) \\ \sin(\theta_0+\theta) \end{bmatrix}. \quad (7)$$

It is obvious that $R(\theta)$ shifts the phase of $|\phi\rangle$. The circuit representation of $R(\theta)$ is shown in Fig.3.

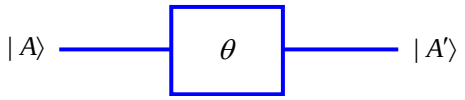

Figure 3. Single qubit rotation gate.

C. Unitary Operators and Tensor Products

A matrix V is said to be unitary if $(V^*)^T V$=I, where the "*" indicates complex conjugation, and "T" indicates the transpose operation, I indicates the unit matrix. Similarly an operator U is unitary if $(U^*)^T U$=I. It is easily checked that an operator is unitary if and only if its matrix representations is unitary.

The *tensor product* is a way of putting vector spaces together to form larger vector spaces. This construction is crucial to understanding the quantum mechanics of multi-particle system. Suppose V and W are vector spaces of dimension m and n, respectively. For convenience we also suppose the V and W are Hilbert spaces. Then $V \otimes W$ (read "V tensor W") is an mn dimensional vector space. The elements of $V \otimes W$ are linear combinations of *tensor products* $|v\rangle \otimes |w\rangle$ of elements $|v\rangle$ of V and $|w\rangle$ of W. In particular, if $|i\rangle$ and $|j\rangle$ are orthonormal bases for the spaces V and W then $|i\rangle \otimes |j\rangle$ is a basis for $V \otimes W$. We often use the abbreviated notations $|v\rangle|w\rangle$, $|v,w\rangle$ or even $|vw\rangle$ for the tensor product $|v\rangle \otimes |w\rangle$. For example, if V is a 2-D vector space with basis vectors $|0\rangle$ and $|1\rangle$ then $|0\rangle \otimes |0\rangle$ and $|1\rangle \otimes |1\rangle$ is an element of $V \otimes V$.

D. Multi-qubits Controlled-NOT Ggate

The prototypical multi-qubit quantum logic gate is the controlled-NOT or CNOT gate. This gate has two input qubits, namely the control qubit and target qubit, respectively. The circuit representation of this gate is given in Fig.4.

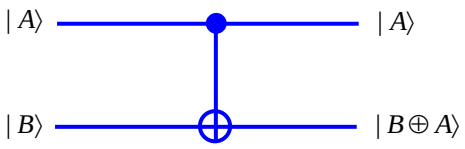

Figure 4. Two-qubit controlled-NOT gate.

The action of this gate can be described as the following. If the control qubit is set to 0, the target qubit is left alone. If the control qubit is set to 1, the target qubit is flipped [12].

In the two-qubit controlled-NOT gate, how to condition on a single qubit set is obvious. This condition can be generalized to the multi-qubits controlled-NOT gate. Suppose we have $n+1$ qubits, and X is a single qubit NOT gate. We define the multi-qubits controlled-NOT operation $C^n(X)$ by

$$C^n(|x_1x_2\cdots x_n\rangle|\phi\rangle) = |x_1x_2\cdots x_n\rangle|X^{x_1x_2\cdots x_n}|\phi\rangle, \quad (8)$$

where $x_1x_2\cdots x_n$ in the exponent of X is the product of the bits $x_1, x_2, \cdots, x_n$. That is, the operator X is applied to the last one qubit (the target qubit), if the first n qubits (the control qubits) are all equal to one. Otherwise, no action is taken. The circuit representation of $C^n(X)$ is shown in Fig.5.

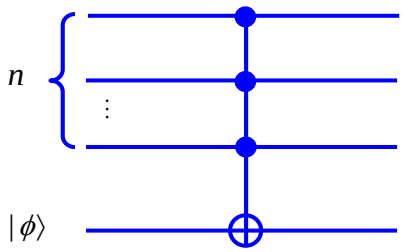

Figure 5. Multi-qubits controlled-NOT gate.

Assume that the control qubits are given by $|x_i\rangle = a_i|0\rangle + b_i|1\rangle$, where $i = 1,2,\cdots,n$. When the target qubit $|\phi\rangle = |0\rangle$, the output of the multi-qubits controlled-NOT gate can be described as

$$\begin{aligned} &C^n(|x_1\rangle \otimes \cdots \otimes |x_n\rangle \otimes |0\rangle) \\ &= |x_1\rangle \otimes \cdots \otimes |x_n\rangle \otimes |0\rangle - b_1b_2\cdots b_n |1\overbrace{1\cdots1}^{n}0\rangle + b_1b_2\cdots b_n |1\overbrace{1\cdots1}^{n}1\rangle \end{aligned} \quad (9)$$

It is clear from Eq.(9) that the output of $C^n(X)$ is in the entangled state of $n+1$ qubits, and the probability of the target qubit state, in which $|1\rangle$ is observed, equals to $(b_1b_2\cdots b_n)^2$.

III. Quantum-Inspired Neural Networks

In this paper, a new Quantum-inspired Neural Networks model (QINN) is proposed, as illustrated in Fig.6, where $|x_1\rangle, |x_2\rangle, \cdots, |x_n\rangle$ are the network input, $|h_1\rangle, |h_2\rangle, \cdots, |h_p\rangle$ are the output of the hidden layer, and $|y_1\rangle, |y_2\rangle, \cdots, |y_m\rangle$ are the output of the output layer.

A. Quantum State Description of Samples

For the l^{th} $\bar{X}^l = [\bar{x}_1^l, \bar{x}_2^l, \cdots, \bar{x}_n^l]^T$ in n-dimension sample Euclid-space, where $l = 1, 2, \cdots, L$, L is the total number of samples, the corresponding quantum state description is defined as

$$|X^l\rangle = [|x_1^l\rangle, |x_2^l\rangle, \cdots, |x_n^l\rangle]^T, \quad (10)$$

where $|x_i^l\rangle = \left[\cos\left(\frac{2\pi(\bar{x}_i^l - \min_i)}{\max_i - \min_i}\right) \quad \sin\left(\frac{2\pi(\bar{x}_i^l - \min_i)}{\max_i - \min_i}\right)\right]^T$,

$\max_i = \max(x_i^1, x_i^2, \cdots, x_i^L)$, $\min x_i = \min(x_i^1, x_i^2, \cdots, x_i^L)$.

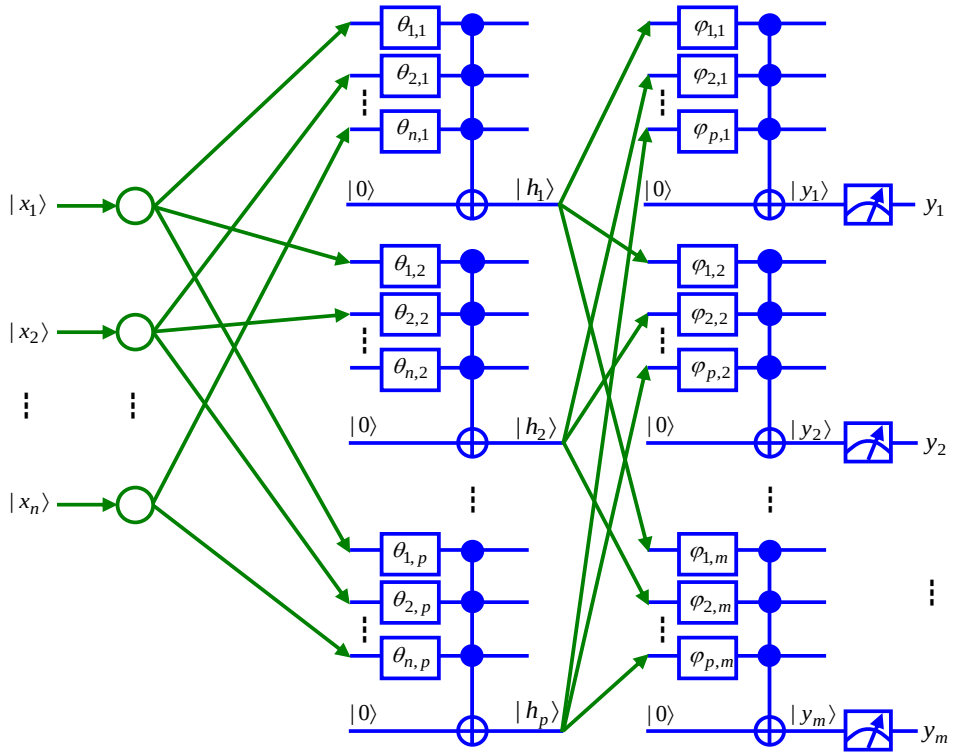

Figure 6. Quantum-inspired neural networks model.

B. Output of Nodes in Every Layer

Let $|x_i\rangle = \cos\theta_i|0\rangle + \sin\theta_i|1\rangle$. According to Eqs. (7)-(9), the output of each layer of our networks can be written as

$$|h_j\rangle = \cos(\varphi_j)|0\rangle + \sin(\varphi_j)|1\rangle, \tag{11}$$

$$|y_k\rangle = \cos(\xi_k)|0\rangle + \sin(\xi_k)|1\rangle, \tag{12}$$

where $i = 1,2,\cdots,n$, $j = 1,2,\cdots,p$, $k = 1,2,\cdots,m$;

$$\varphi_j = \arcsin\left(\prod_{i=1}^{n}\sin(\theta_i + \theta_{ij})\right), \xi_k = \arcsin\left(\prod_{j=1}^{p}\sin(\varphi_j + \varphi_{jk})\right).$$

In this paper, we define the output of the nodes in each layer as the probability amplitude of the corresponding state, in which $|1\rangle$ is observed. Thus, the actual output of our networks is rewritten as follows

$$h_j = \prod_{i=1}^{n}\sin(\theta_i + \theta_{ij}), \tag{13}$$

$$y_k = \prod_{j=1}^{p}\sin\left(\arcsin\left(\prod_{i=1}^{n}\sin(\theta_i + \theta_{ij})\right) + \varphi_{jk}\right). \tag{14}$$

C. Parameters Update in Every Layer

In this QINN model, the parameters to be updated are the rotation angles of the quantum rotation gates in every layer. Suppose the desired normalized outputs are $\tilde{y}_1, \tilde{y}_2, \cdots, \tilde{y}_m$. The evaluation function is defined as

$$E = \max_{1\le l\le L}\max_{1\le k\le m}|e_k^l| = \max_{1\le l\le L}\max_{1\le k\le m}|\tilde{y}_k^l - y_k^l|. \tag{15}$$

According to the gradient descent algorithm, the gradient of the rotation angles in every layer can be calculated as follows

$$\frac{\partial e_k^l}{\partial \theta_{ij}} = -y_k ctg(\varphi_j + \varphi_{jk}) h_j^l ctg(\theta_i + \theta_{ij}) / \sqrt{1-(h_j^l)^2}, \tag{16}$$

where $h_j^l = \prod_{i=1}^{n}\sin(\theta_i^l + \theta_{ij})$.

$$-\frac{\partial e_k^l}{\partial \varphi_{jk}} = -y_k ctg(\varphi_j + \varphi_{jk}), \tag{17}$$

Because gradient calculation is more complicated, the standard gradient descent algorithm is not easy to convergence. Hence we employ the L-M algorithm to adjust the QINN parameters.

Let p denote the parameter vector, e denote the error vector, and J denote the Jacobian matrix. p, e and J are respectively defined as follows

$$P^{\mathrm{T}} = [\theta_{1,1}, \theta_{1,2}, \cdots, \theta_{n,p}, \varphi_{1,1}, \varphi_{1,2}, \cdots, \varphi_{p,m}], \quad (18)$$

$$e^{\mathrm{T}}(p) = [e_1^1, e_2^1, \cdots, e_m^1, \cdots, e_1^L, e_2^L, \cdots, e_m^L], \quad (19)$$

$$J(P) = \begin{bmatrix} \frac{\partial e_1^1}{\partial \theta_{1,1}} & \cdots & \frac{\partial e_1^1}{\partial \theta_{n,p}} & \cdots & \frac{\partial e_1^1}{\partial \varphi_{1,1}} & \cdots & \frac{\partial e_1^1}{\partial \varphi_{p,m}} \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \frac{\partial e_m^1}{\partial \theta_{1,1}} & \cdots & \frac{\partial e_m^1}{\partial \theta_{n,p}} & \cdots & \frac{\partial e_m^1}{\partial \varphi_{1,1}} & \cdots & \frac{\partial e_m^1}{\partial \varphi_{p,m}} \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \frac{\partial e_1^L}{\partial \theta_{1,1}} & \cdots & \frac{\partial e_1^L}{\partial \theta_{n,p}} & \cdots & \frac{\partial e_1^L}{\partial \varphi_{1,1}} & \cdots & \frac{\partial e_1^L}{\partial \varphi_{p,m}} \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ \frac{\partial e_m^L}{\partial \theta_{1,1}} & \cdots & \frac{\partial e_m^L}{\partial \theta_{n,p}} & \cdots & \frac{\partial e_m^L}{\partial \varphi_{1,1}} & \cdots & \frac{\partial e_m^L}{\partial \varphi_{p,m}} \end{bmatrix}. \quad (20)$$

According to L-M algorithm, the QINN iterative equation is written as follows

$$P_{t+1} = P_t - (J^{\mathrm{T}}(P_t)J(P_t) + \mu_t I)^{-1} J^{\mathrm{T}}(P_t)e(P_t) \quad (21)$$

where t denotes the iterative steps, I denotes the unit matrix, and μ_t is a small positive number to ensure the matrix $J^{\mathrm{T}}(P_t)J(P_t) + \mu_t I$ invertible.

If the value of the evaluation function E reaches the predefined precision within the preset maximum number of iterative steps, then the execution of the algorithm is stopped, else the algorithm is not stopped until it reaches the predefined maximum number of iterative steps.

D. Multiple Attractors Theorems

It can be seen from Eq.(14) that there exist many global optimum solutions (attractors) for the iteration sequences $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$. In other words, the global optimal solution is not the only one. In fact, we have the following theorems.

Theorem 1. If $\tilde{\theta}_{ij}$ and $\tilde{\varphi}_{jk}$ are the global optimum solutions of the iteration sequences $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$, then $\bar{\theta}_{ij}$ and $\bar{\varphi}_{jk}$ defined by Eqs.(22)-(23) are also the global optimum solutions of $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$, where n_1 and n_2 are arbitrary integers.

$$\begin{cases} \bar{\theta}_{ij} = 2n_1\pi + \tilde{\theta}_{ij} \\ \bar{\varphi}_{jk} = 2n_2\pi + \tilde{\varphi}_{jk} \end{cases} \quad (22)$$

$$\begin{cases} \bar{\theta}_{ij} = (2n_1+1)\pi - (2\theta_i + \tilde{\theta}_{ij}) \\ \bar{\varphi}_{jk} = 2n_2\pi + \tilde{\varphi}_{jk} \end{cases} \quad (23)$$

Proof. Suppose y_k is the actual output corresponding to the desired output $\tilde{y}_k$. According to Eq.(14), we have

$$\begin{aligned} y_k &= \prod_{j=1}^{p} \sin(\arcsin(\prod_{i=1}^{n} \sin(\theta_i + 2n_1\pi + \tilde{\theta}_{ij})) + 2n_2\pi + \tilde{\varphi}_{jk}) \\ &= \prod_{j=1}^{p} \sin\left(\arcsin\left(\prod_{i=1}^{n} \sin\left(\theta_i + \tilde{\theta}_{ij}\right)\right) + \tilde{\varphi}_{jk}\right) = \tilde{y}_k \end{aligned} \quad . (24)$$

Hence, $\bar{\theta}_{ij}$ and $\bar{\varphi}_{jk}$ defined by Eq.(22) are the global optimum solutions of $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$.

$$\begin{aligned} y_k &= \prod_{j=1}^{p} \sin(\arcsin(\prod_{i=1}^{n} \sin(\theta_i + (2n_1+1)\pi - (2\theta_i + \tilde{\theta}_{ij}))) + 2n_2\pi + \tilde{\varphi}_{jk}) \\ &= \prod_{j=1}^{p} \sin(\arcsin(\prod_{i=1}^{n} \sin(\pi - (\theta_i + \tilde{\theta}_{ij}))) + \tilde{\varphi}_{jk}) \\ &= \prod_{j=1}^{p} \sin(\arcsin(\prod_{i=1}^{n} \sin(\theta_i + \tilde{\theta}_{ij})) + \tilde{\varphi}_{jk}) = \tilde{y}_k \end{aligned} \quad .(25)$$

Hence, the $\bar{\theta}_{ij}$ and $\bar{\varphi}_{jk}$ defined by Eq.(23) are the global optimum solutions of $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$. □

Theorem 2. For $\forall \phi \in R$, there exist all the global optimum solutions of the iteration sequences $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$ in range $[\phi, \phi + 2\pi]$, where R denotes the real number set.

Proof. Suppose $\tilde{\theta}_{ij}$ and $\tilde{\varphi}_{jk}$ are the global optimum solutions of the iteration sequences $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$, and $\tilde{\theta}_{ij} \notin [\phi, \phi + 2\pi]$, $\tilde{\varphi}_{jk} \notin [\phi, \phi + 2\pi]$. There exist two integers n_1 and n_2, which meet $\begin{cases} \bar{\theta}_{ij} = 2n_1\pi + \tilde{\theta}_{ij} \\ \bar{\varphi}_{jk} = 2n_2\pi + \tilde{\varphi}_{jk} \end{cases}$ and $\begin{cases} \bar{\theta}_{ij} \in [\phi, \phi + 2\pi] \\ \bar{\varphi}_{jk} \in [\phi, \phi + 2\pi] \end{cases}$. According to Theorem 1, $\bar{\theta}_{ij}$ and $\bar{\varphi}_{jk}$ are also the global optimum solutions of $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$. Hence, for $\forall \phi \in R$, there exist all the global optimum solutions of the iteration sequences $\{\theta_{ij}(t)\}$ and $\{\varphi_{jk}(t)\}$ in range $[\phi, \phi + 2\pi]$. □

E. Algorithm Description

The structure of proposed algorithms is shown in the following.

Procedure QINN
Begin
  $t \leftarrow 0$
  (1) Initialization of parameters, including:
    a) the predefined precision ε,
    b) the predefined maximum of iterative steps G,

c) the parameter of L-M algorithm μ_t,

d) the parameters of QINN $\theta_{i,j}, \varphi_{j,k} \in (-\pi/2, \pi/2)$.

(2) While (not termination-condition)

Begin

a) computing the actual outputs of all samples by Eqs.(13, 14),

b) computing the value of the evaluation function E by Eq.(15).

c) adjusting the parameters $\theta_{i,j}, \varphi_{j,k}$ by Eq.(21).

d) $t \leftarrow t+1$.

End

End

IV. PRACTICAL APPLICATION

In this section , three examples are used to compare it with the BPNN. Our QINN has the same structure and parameters as the BPNN in the simulations, and the L-M algorithm is also applied in the BPNN.

A. Modeling of Acrylamide Homogeneous Polymerization

Some experimental results for the acrylamide homogeneous polymerization are listed in Table 1.

TABLE I. THE EXPERIMENTAL RESULTS OF POLYMERIZATION CHEMICAL REACTION

t/min	T1/°C	T2/°C	T3/°C	T4/°C	T5/°C	T6/°C	T7/°C	T8/°C	T9/°C
0	17	18	18	18	14	15	15	14	13
10	19	16	21	18	15	16	16	15	15
20	23	16	23	19.5	17	16	18	16	16
30	27	17	26	21.5	20	16.5	21	17.5	18
40	30	18	31	24.5	22	18	25	19.5	19
50	34	19	36	29.5	25	20	30	22	21
60	39	20	43.5	34	28	22	36	24	23
70	45	21	52.5	38	33	24	40	26.5	25
80	52	22.5	59	46	38	29	54	30.5	27
90	64	24	68	56	46	32	64	35.5	30
100	73	26	73	63	52	36	73	43	34
110	81	28	77	74	60	40	76	54.5	41
120	82	32	79	78	66	44.5	77	67	44
130	84	36	83	93	72	50.5	78	79.5	49
140	88	45	87	95	76	57.5	79	83	54
150	92	54	96	97	81	66.5	84	87	65
160	101	64	98	100	83	67	86	93	77
170	113	68	102	110	94	69	89	98	80
Nm	1811	1838	1894	1853	1496	1701	1754	1915	1644

In the Table 1, t denotes the cumulative time by minute starting from 0, T_i (i=1,2,⋯,9) are the testing results of the ith group experimental temperature (°C), Nm are the molecules of resultant ($\times 10^6 cm^3$), known as the concentration of the acrylamide homogeneous polymerization resultant. In Fig.7, we give nine curves that describe the acrylamide homogeneous polymerization experimental temperature varying with time (min). Each curve corresponds to the molecules of a group of experimental resultants. For ease of comparison, some relevant concepts are defined as follows.

Approximation error. Suppose $\bar{y}_k^l$ and y_k^l denote the desired output and actual output after training, respectively. The approximation error is defined as

$$E = \max_{1 \le l \le L} \max_{1 \le k \le m} |\bar{y}_k^l - y_k^l|, \tag{26}$$

where L denotes the total number of the training samples.

Average approximation error. Suppose $E_1, E_2, \cdots, E_N$ denotes the approximation error over N simulations, respectively. The average approximation error is defined as

$$E_{avg} = \frac{1}{N} \sum_{i=1}^{N} E_i . \tag{27}$$

Convergence. Suppose E denotes the approximation error after training, and ε denotes the predefined target error. If $E < \varepsilon$, the networks training is considered to have converged.

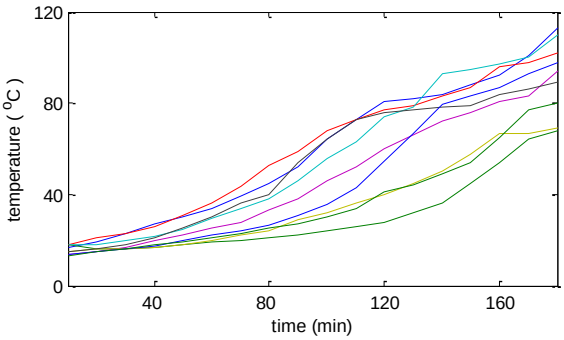

Figure 7. Curves of temperature-time.

For a neural network, each curve corresponds to an input sample, and the number of the resultant molecules is regarded as the corresponding desired output.

In this experiment we apply respectively QINN and BPNN to approximate the data in the Table 1. The L-M algorithm is also employed to adjust weights of the BPNN. There are 9 samples in Table 1, each sample consists of eighteen input values and one output value. Hence, there is eighteen input nodes and one output node in QINN and BPNN. In order to fully compare the approximation ability of two NNs, the number of hidden nodes are respectively set to 5, 6, ⋯ , 20. The normalized maximum target error is set to 10^{-3}, and the maximum number of iterative steps is set to 1000. The QINN rotation angles are initialized to random numbers in $(-\pi/2, \pi/2)$. In BPNN, all weights are initialized to random values in (-1,1). The parameter μ_t of L-M algorithm is 0.01.

To reduce the randomness of contrast results, for each kind of setting of hidden nodes, two NNs are independently trained 10 times, respectively, and their average training results are recorded. Then we use three indicators, such as the average approximation error, the average iterative steps, and the convergence times, to compare QINN and BPNN. Training

result contrasts are shown in Table 2. The same results also are illustrated in Figs.8-9.

TABLE II. TRAINING RESULTS COMPARISON OF QINN AND BPNN

Hidden nodes	average approximation error		average iterative steps		convergence times	
	QINN	*BPNN*	*QINN*	*BPNN*	*QINN*	*BPNN*
5	0.44	0.50	23	66	10	10
6	0.45	0.47	19	38	10	10
7	0.45	0.49	19	37	10	10
8	0.43	0.49	20	37	10	10
9	0.42	0.49	16	37	10	10
10	0.41	0.48	16	34	10	10
11	0.40	0.50	14	38	10	10
12	0.40	0.49	15	35	10	10
13	0.43	0.50	18	36	10	10
14	0.39	0.49	14	37	10	10
15	0.41	0.49	13	36	10	10
16	0.44	0.50	14	36	10	10
17	0.41	0.49	30	39	10	10
18	0.43	0.49	14	36	10	10
19	0.39	0.48	14	37	10	10
20	0.42	0.49	12	36	10	10

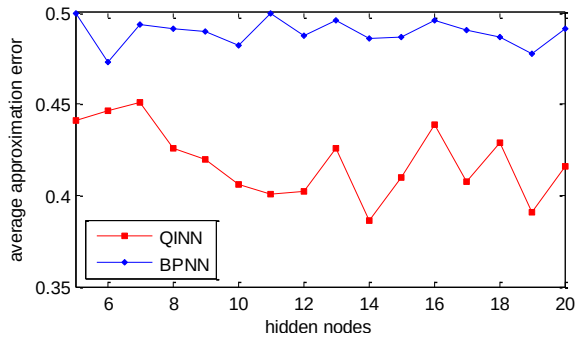

Figure 8. The average approximation error contrast.

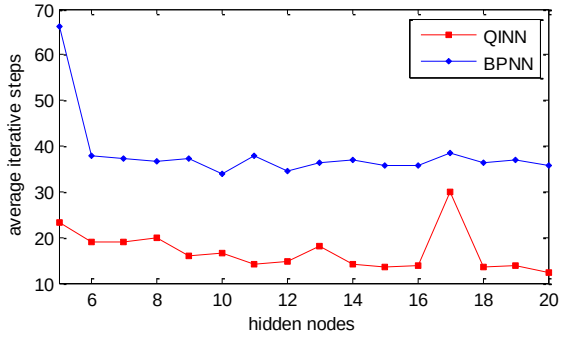

Figure 9. The average iterative steps contrast.

Form the experimental results, as far as the average approximation error is concerned, that of QINN is obviously less than that of BPNN under each setting of hidden nodes. Although both QINN and BPNN have converged in all experiments, the average iterative steps of QINN are obviously less than that of BPNN. The experimental results show that the QINN is obviously superior to the BPNN.

B. Wine Recognition

In this section, we apply QINN to identify the type of wine. All the instances data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The data comes from the following url: http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv.

All instances are divided into three categories, marked as 1, 2, 3 respectively. Each of instances contains 13 continuous attributes as follows: 1) Alcohol; 2) Malic acid; 3) Ash; 4) Alcalinity of ash; 5) Magnesium; 6) Total phenols; 7) Flavanoids; 8) Nonflavanoid phenols; 9) Proanthocyanins; 10) Color intensity; 11) Hue; 12) OD280/OD315 of diluted wines; 13) Proline. The number of instances per class is equal to 48.

According to the characteristics of instances, we know that both QINN and BPNN have thirteen input nodes, one output node. In order to enhance the objectivity of comparison results, we set the number of hidden nodes equal to 10, 11, ⋯, 20, respectively. The normalized maximum absolute error is set to 1/6, and the maximum number of iterative steps is set to 100. The other parameters of two NNs are set in the same way as the previous simulation. For ease of comparison, some evaluation indicators are defined as follows.

The number of correct recognition. Suppose $\bar{y}^1, \bar{y}^2, \cdots, \bar{y}^M$ denote the desired outputs of M samples, and $y^1, y^2, \cdots, y^M$ denote the corresponding actual outputs, where M denotes the number of samples in training set. The number of correct recognition for training set is defined as

$$N_{tr} = \sum_{n=1}^{N} (M - \sum_{m=1}^{M} |\bar{y}^m - |[y^m]|) / N \tag{28}$$

where N denotes the total number of training trials, "$[x]$" denote the integer closest to the real number x, where by convention we round halves down. Similarly, the number of correct recognition for the testing set is defined as

$$N_{te} = \sum_{n=1}^{N} (\overline{M} - \sum_{m=1}^{\overline{M}} |\bar{y}^m - |[y^m]|) / N \tag{29}$$

where M denotes the number of samples in testing set.

The ratio of correct recognition. The ratio of correct recognition for the training set is defined as

$$R_{tr} = 100 N_{tr} / M \tag{30}$$

Similarly, the ratio of correct recognition for the testing set is defined as

$$R_{te} = 100 N_{te} / \overline{M} \tag{31}$$

In all samples, we use the first 96 data to train networks, use the remaining 48 data to test the generalization of two NNs, and use five indicators (namely, the ratio of correct recognition for training set and testing set, the average iterative steps, the average running time T_{avg}, and the convergence times) to compare QINN with BPNN. To reduce the randomness of contrast results, for each kind of setting of hidden nodes, two NNs are independently trained 100 times, respectively, and their average training results are recorded. The training result contrasts are shown in Tables 3,4, and the same results also are illustrated in Figs.10-12.

TABLE III. RECOGNITION RATIO COMPARISON FOR WINE RECOGNITION

Hidden nodes	ratio of correct recognition for training		ratio of correct recognition fro testing	
	QINN	*BPNN*	*QINN*	*BPNN*
10	100	100	97.52	97.17
11	100	100	97.48	96.79
12	100	100	97.85	96.94
13	100	100	97.75	96.96
14	100	100	98.29	96.75
15	100	100	97.81	97.06
16	100	100	97.85	96.54
17	100	100	98.21	96.33
18	100	100	98.23	96.94
19	100	100	98.19	96.56
20	100	100	98.60	96.23

TABLE IV. PERFORMANCE COMPARISON OF QINN AND BPNN

Hidden nodes	average running time (s)		average iterative steps		convergence times	
	QINN	*BPNN*	*QINN*	*BPNN*	*QINN*	*BPNN*
10	1.176	1.190	29.83	52.28	100	100
11	1.224	1.344	28.01	52.34	100	99
12	1.274	1.499	26.53	51.72	100	100
13	1.395	1.674	26.33	52.18	100	100
14	1.513	1.830	25.65	51.17	100	100
15	1.640	2.067	25.32	51.73	100	100
16	1.703	2.237	24.48	50.84	100	100
17	1.825	2.471	24.74	50.26	100	100
18	1.993	2.752	23.79	50.90	100	100
19	2.193	2.945	24.21	49.41	100	100
20	2.328	3.424	23.23	50.98	100	100

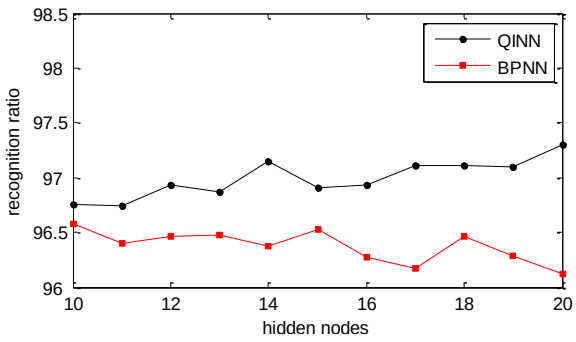

Figure 10. Recognition ratio contrasts of QINN and BPNN.

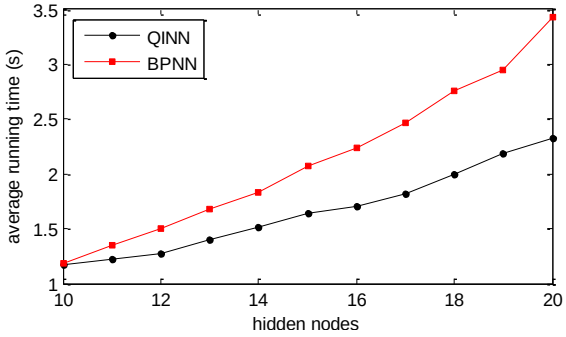

Figure 11. Average running time contrasts of QINN and BPNN.

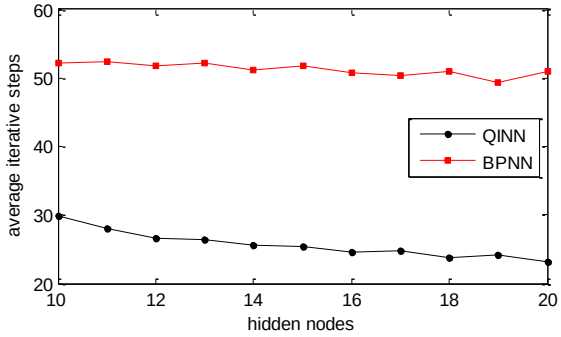

Figure 12. Average iterative steps contrasts of QINN and BPNN.

From Table 3 and Fig 10, although both QINN and BPNN obtain the recognition ratio of 100% for training set, QINN obtains the greater recognition ratio than BPNN for testing set. On the other hand, From Table 4 and Fig 11-12, although the average running time of QINN is slightly less than that of BPNN, the average iterative steps of QINN is obviously less than that of BPNN. Experimental results show that the QINN has a better approximation and generalization capability than the BPNN.

C. Time series prediction for Mackey-Glass

Mackey-Glass time series can be generated by the following iterative equation

$$x(t+1) - x(t) = a\frac{x(t-\tau)}{1+x^{10}(t-\tau)} - bx(t), \tag{32}$$

where t and τ are integers, $a = 0.2$, $b = 0.1$, $\tau = 17$, and $x(0) \in (0,1)$.

From the above equation, we may obtain the time sequence $\{x(t)\}_{t=1}^{1000}$, We take the first 800, namely $\{x(t)\}_{t=1}^{800}$, as the training set, and the remaining 200, namely $\{x(t)\}_{t=801}^{1000}$, as the testing set. Our prediction scheme is to employ 15 data adjacent to each other to predict the next one data. Namely, in our model, the number of input nodes equals to15, and there is only one output node.

In order to fully compare the approximation ability of two models, the number of hidden nodes is respectively set to $10, 15, 20, 25, 30$. The predefined precision is set to 0.05, and the maximum of iterative steps is set to 100. The QINN rotation angles are initialized to random numbers in $(-\pi/2, \pi/2)$. For BPNN, all weights are initialized to random numbers in (-1,1), and the *Sigmoid* functions are used to activation functions in hidden layer and output layer. Some relevant concepts are defined as follows.

Approximation error. Suppose $\bar{y}^l$ and y^l denote desired output and the corresponding actual output after training, respectively. The approximation error is defined as $E = \max_{1\le l\le L} |\bar{y}^l - y^l|$. where L denotes the number of the training samples.

Average approximation error. Suppose $E_1, E_2, \cdots, E_N$ denote the approximation error over N experiments, respectively. The average approximation error E_{avg} is defined as the arithmetic mean of these $E_1, E_2, \cdots, E_N$.

Convergence ratio. Suppose E denotes the approximation error after training, and ε denotes the target error. If $E < \varepsilon$, the network training is considered to have converged. Suppose N denotes the total number of training trials, and C denotes the number of convergent training trials. The convergence ratio is defined as $\lambda = C/N$.

Iterative steps. In a training trials, the times of adjusting all network parameters is defined as iterative steps.

Average iterative steps. Suppose $S_1, S_2, \cdots, S_N$ denote the iterative steps over N training trials, respectively. The average iterative steps S_{avg} is defined as the arithmetic mean of these $S_1, S_2, \cdots, S_N$.

Average running time. Suppose $T_1, T_2, \cdots, T_N$ denote the running time over N training trials, respectively. The average running time T_{avg} is defined as the arithmetic mean of these $T_1, T_2, \cdots, T_N$.

Our experiment scheme is that, for each of hidden nodes, QINN and BPNN are respectively run 10 times. Then we use four indicators, such as *the average approximation error*, *the average iterative steps*, *the average running time*, and *the convergence ratio*, to compare QINN with BPNN. Training result contrasts are shown in Tables 5.

Form Tables 5, we can see that the performance of QINN is obviously superior to that of BPNN, and the QINN has better stability than BPNN when the number of hidden nodes changes.

Next, we investigate the generalization ability of QINN. Our experiment scheme is that, the QINN and BPNN are respectively done 10 training on the training set, and are immediately investigated the generalization ability on the testing set after each training. The average results of 10 testing are regarded as the evaluation indexes. To facilitate comparison, we first present the following definition of evaluation indexes.

TABLE V. TRAINING RESULTS COMPARISON OF QINN AND BPNN

Index	model	Hidden nodes				
		10	*15*	*20*	*25*	*30*
E_{avg}	QINN	0.0426	0.0420	0.0419	0.0418	0.0428
	BPNN	0.3198	0.1377	0.2292	0.4122	0.2284
S_{avg}	QINN	6.20	6.10	6.30	4.90	5.90
	BPNN	45.8	23.9	33.4	47.6	29.8
T_{avg} (s)	QINN	16.65	19.21	28.15	29.04	35.48
	BPNN	23.02	21.73	42.95	63.37	74.48
λ (%)	QINN	100	100	100	100	100
	BPNN	70	90	80	60	80

Average prediction error. Suppose $\bar{y}^l$ and $\hat{y}^l(t)$ denote the desired output and the corresponding prediction output after the t^{th} testing respectively. The average prediction error over N testing is defined as $\bar{E}_{avg} = \frac{1}{N}\sum_{t=1}^{N} \max_{1\le l\le L} |\bar{y}^l - \hat{y}^l(t)|$.

Average error mean. Suppose $\bar{y}^l$ and $\hat{y}^l(t)$ denote the desired output and the corresponding prediction output after the t^{th} testing respectively. The average error mean over N testing is defined as $\bar{E}_{mean} = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{L}\sum_{l=1}^{L}|\bar{y}^l - \hat{y}^l(t)|$.

The evaluation indexes contrast of QINN and BPNN are shown in Table 6. The experimental results show that the generalization ability of QINN is obviously superior to that of BPNN.

TABLE VI. PREDICTION RESULTS COMPARISON OF QINN AND BPNN

Index	model	Hidden nodes				
		10	*15*	*20*	*25*	*30*
$\bar{E}_{avg}$	QINN	0.051	0.050	0.051	0.051	0.053
	BPNN	0.332	0.152	0.244	0.425	0.243
$\bar{E}_{mean}$	QINN	0.009	0.009	0.009	0.009	0.009
	BPNN	0.148	0.053	0.107	0.182	0.106

Finally, we theoretically explain the above experimental results. Assume that *n* denotes the number of input nodes, *p* denotes the number of hidden nodes, and *m* denotes the number of output nodes. It is clear that the number of adjustable parameters in QINN and BPNN is the same, ie, equals *np+pm*. However, the methods of adjusting parameters are completely different. For processing of input information, QINN and BPNN take the different approaches. In QINN, using quantum information processing mechanism, the inputs are circularly mapped to the output of quantum controlled-NOT gates in hidden layer and output layer. As the controlled-NOT gate's output is in the entangled state of multi-qubits, therefore, this mapping is highly nonlinear, which make QINN have the stronger approximation and generalization ability, which is

consistent with the experimental results. On the other hand, the number of the attractors of the QINN is more than that of the BPNN, and the multiple attractors are the main feature of the QINN. Actually, compared with the BPNN, the existence of the inbuilt periodic attractors in the QINN results in an efficient convergence, which is consistent with the conclusions of Theorem 2.

It is worth pointing out that QINN is potentially much more computationally efficient than all the models referenced above in the Introduction section. The efficiency of many quantum algorithms comes directly from quantum parallelism that is a fundamental feature of many quantum algorithms. Heuristically, and at the risk of over-simplifying, quantum parallelism allows quantum computers to evaluate a function $f(x)$ for many different values of x simultaneously. Although quantum simulation requires many resources in general, quantum parallelism leads to very high computational efficiency by using the superposition of quantum states. In QINN, the input samples have been converted into corresponding quantum superposition states after preprocessing. Hence, as far as a lot of quantum rotation gates and controlled-NOT gates are concerned in QINN, information processing can be performed simultaneously, which greatly improves the computational efficiency. Because the above two experiments are performed in classical computer, the quantum parallelism has not been explored. However, the efficient computational ability of QINN is bound to stand out in future quantum computer.

V. CONCLUSIONS

This paper proposes quantum-inspired neural network model based on the principle of quantum computing. The architecture of the proposed model includes three layers, where both the hidden layer and output layer consists of quantum neurons. An obvious difference from classical BPNN is that each dimension of a single input sample consists a qubit rather that a value. The activation function of hidden layer and output layer are redesigned according to the principle of quantum computing. The L-M algorithm is employed for learning. With application of the information processing mechanism of quantum controlled-NOT gates, proposed model can effectively obtain the sample characteristics. The experimental results reveal that the approximation and generalization abilities of proposed model are obviously stronger than that of the BPNN. The following issues of the proposed model, such as continuity, computational complexity, and improvement of learning algorithm, are subject of further research.

REFERENCES

[1] V. Nekoukar, M. Taghi, "A local linear radial basis function neural network for financial time-series forecasting", Application Intelligence, 2010, 33, pp. 352-356.

[2] H. Lee, E. Kim, W. Pedrycz, "A new selective neural network ensemble with negative correlation", Application Intelligence, 2012, 37, pp. 488-498.

[3] B.J. Park, W. Pedrycz, S.K. Oh, "Polynomial-based radial basis function neural networks (P-RBF NNs) and their application to pattern classification", Application Intelligence, 2010, 32, pp. 27-46.

[4] Y. F. Hassan, "Rough sets for adapting wavelet neural networks as a new classifier system", Application Intelligence, 2011, 35, pp. 260-268.

[5] K.J. Kim, H. Ahn, "Simultaneous optimization of artificial neural networks for financial forecasting", Application Intelligence, 2012, 36, pp. 887-898.

[6] S. Kak, "On quantum neural computing," Information Sciences, 1995, 83, pp. 143-160.

[7] P. Gopathy, B.K. Nicolaos, "Quantum Neural networks (QNN's): Inherently fuzzy feedforward neural networks," IEEE Transactions on Neural Networks, 1997, 8 (3), pp. 679-693.

[8] N. Matsui, M. Takai, H. Nishimura, "A network model based on qubit-like neuron corresponding to quantum circuit," Trans. IEICE, J81-A (1998), pp. 1687-1692.

[9] N. Matsui, N. Kouda, H. Nishimura, "Neural network based on QBP and its performance," Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural networks. 2000, 3 (24-27), pp. 247-252.

[10] F. Shafee, "Neural networks with quantum gated nodes," Engineering Applications of Artificial Intelligence. 2007, 20(4), pp. 429-437.

[11] P.C. Li, S.Y. Li, "Learning algorithm and application of quantum BP neural networks based on universal quantum gates," Journal of Systems Engineering and Electronics. 2008, 19(1), pp. 167-174.

[12] M.A. Nielsen, M. Chuang, "Quantum Computation and quantum Information," London: Cambridge University Press, 2000.