Appl Intell (2014) 40:107–126
DOI 10.1007/s10489-013-0447-3

Model and algorithm of quantum-inspired neural network with sequence input based on controlled rotation gates

Panchi Li · Hong Xiao

Published online: 30 May 2013

Abstract To enhance the approximation and generalization ability of classical artificial neural network (ANN) by employing the principles of quantum computation, a quantum-inspired neuron based on controlled-rotation gate is proposed. In the proposed model, the discrete sequence input is represented by the qubits, which, as the control qubits of the controlled-rotation gate after being rotated by the quantum rotation gates, control the target qubit for rotation. The model output is described by the probability amplitude of state $|1\rangle$ in the target qubit. Then a quantum-inspired neural network with sequence input (QNNSI) is designed by employing the quantum-inspired neurons to the hidden layer and the classical neurons to the output layer. An algorithm of QNNSI is derived by employing the Levenberg–Marquardt algorithm. Experimental results of some benchmark problems show that, under a certain condition, the QNNSI is obviously superior to the ANN.

Keywords Quantum computation · Quantum rotation gate · Controller-rotation gate · Quantum-inspired neuron · Quantum-inspired neural network

1 Introduction

Over the last few decades, many researchers 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 back propagation algorithm [5].

Many neurophysiological experiments indicate that the information processing character of the biological nerve system mainly includes the following eight aspects: the spatial aggregation, the multi-factor aggregation, the temporal cumulative effect, the activation threshold characteristic, self-adaptability, exciting and restraining characteristics, delay characteristics, conduction and output characteristics [6]. From the definition of the M–P neuron model, classical ANN preferably simulates voluminous biological neurons' characteristics such as the spatial weight aggregation, self-adaptability, conduction and output, but it does not fully incorporate temporal cumulative effect because the outputs of ANN depend only on the inputs at the moment regardless of the prior moment. In the process of practical information processing, the memory and output of the biological neuron not only depend on the spatial aggregation of each input information, but also are related to the temporal cumulative effect. Although the ANNs in Refs. [7–10] can process temporal sequences and simulates delay characteristics of biological neurons; in these models, the temporal cumulative effect has not been fully reflected. Traditional ANN

P. Li (✉) · H. Xiao
School of Computer & Information Technology,
Northeast Petroleum University, Daqing 163318, China
e-mail: lipanchi@vip.sina.com

can only simulate point-to-point mapping between the input space and output space. A single sample can be described as a vector in the input space and output space. However, the temporal cumulative effect denotes that multiple points in the input space are mapped to a point in the output space. A single input sample can be described as a matrix in the input space, and a single output sample is still described as a vector in the output space. In this case, we claim that the network has a sequence input.

Since Kak [11] firstly proposed the concept of quantum-inspired neural computation in 1995, quantum neural network (QNN) has attracted a great attention by the international scholars during the past decade, and a large number of novel techniques have been studied for quantum computation and neural network. For example, Purushothaman et al. [12] proposed the model of quantum neural network with multilevel hidden neurons based on the superposition of quantum states in the quantum theory. In Ref. [13], an attempt was made to reconcile the linear reversible structure of quantum evolution with nonlinear irreversible dynamics of neural network. Michiharu et al. [14] presented a novel learning model with qubit neuron according to quantum circuit for XOR problem and describes the influence to learning by reducing the number of neurons. In Ref. [15], a new mathematical model of quantum neural network was defined, building on Deutsch's model of quantum computational network, which provides an approach for building scalable parallel computers. Fariel Shafee [16] proposed the neural network with the quantum gated nodes, and indicates that such quantum network may contain more advantageous features from the biological systems than the regular electronic devices. In our previous work [17], we proposed a quantum BP neural network model with learning algorithm based on the single-qubit rotation gates and two-qubits controlled-rotation gates. In Ref. [18], we proposed a neural network model with quantum gated nodes and a smart algorithm for it, which shows superior performance in comparison with a standard error back propagation network. Adenilton et al. [19] proposed a weightless model based on quantum circuit. It is not only quantum-inspired but is actually a quantum NN. This model is based on Grover's search algorithm, and it can perform both quantum learning and simulate the classical models. However, all the above QNN models, like M–P neurons, it also does not fully incorporate temporal cumulative effect because a single input sample is either irrelative to time or relative to a moment instead of a period of time.

In this paper, in order to fully simulate biological neuronal information processing mechanisms and to enhance the approximation and generalization ability of ANN, we proposed a qubit neural network model with sequence input based on controlled-rotation gates, called QNNSI. It's worth pointing out that an important issue is how to define, configure and optimize artificial neural networks. Refs. [20, 21] make a deep research into this question. After repeated experiments, we opt to use a three-layer model with a hidden layer, which employs the Levenberg–Marquardt algorithm for learning. Under the premise of considering approximation ability and computational efficiency, this option is a relatively ideal. The proposed approach is utilized to predict the year mean of sunspot number, and the experimental results indicate that, under a certain condition, the QNNSI is obviously superior to the common ANN.

2 The qubit and quantum gate

2.1 Qubit

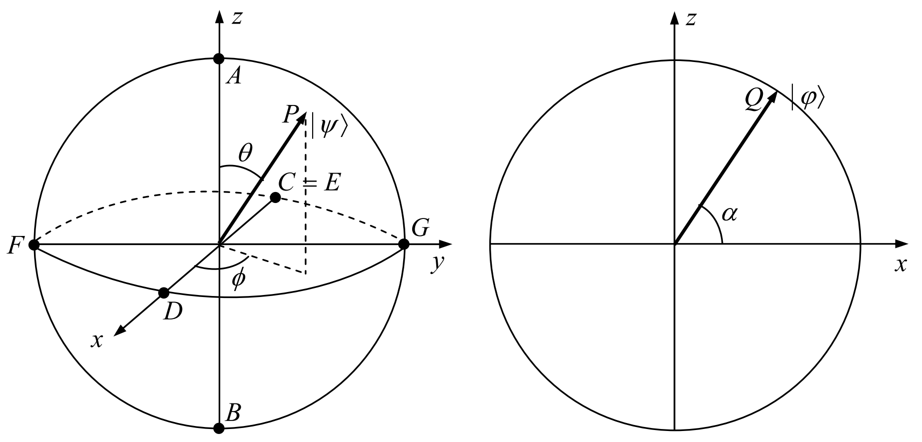
What is a qubit? Just as a classical bit has a state-either 0 or 1—a qubit also has a state. Two possible states for a qubit are the state $|0\rangle$ and $|1\rangle$, which as you might guess correspond to the states 0 and 1 for a classical bit. Notation like $|\ \rangle$ is called the *Dirac* notation,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

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

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 continuum 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(a), where the points corresponding to the following states are shown: $|A\rangle = [1, 0]^{\mathrm{T}}$, $|B\rangle = [0, 1]^{\mathrm{T}}$, $|C\rangle = |E\rangle = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]^{\mathrm{T}}$, $|D\rangle = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^{\mathrm{T}}$, $|F\rangle = [\frac{1}{\sqrt{2}}, -\frac{i}{\sqrt{2}}]^{\mathrm{T}}$, $|G\rangle = [\frac{1}{\sqrt{2}}, \frac{i}{\sqrt{2}}]^{\mathrm{T}}$. For convenience, in this paper, we represent the qubit's state by a point on a circle of unit radius as shown in Fig. 1(b). The corresponding

Fig. 1 A qubit description



(a) A qubit description on Bloch sphere (b) A qubit description on unit circle

relations between Figs. 1(a) and 1(b) can be written as

$$\begin{cases} \alpha : 0 \longrightarrow \pi/2 & \Longleftrightarrow \quad \phi = 0 \text{ and } \theta : \pi/2 \longrightarrow 0, \\ \alpha : \pi/2 \longrightarrow \pi & \Longleftrightarrow \quad \phi = \pi \text{ and } \theta : 0 \longrightarrow \pi/2, \\ \alpha : \pi \longrightarrow 3\pi/2 & \Longleftrightarrow \quad \phi = \pi \text{ and } \theta : \pi/2 \longrightarrow \pi, \\ \alpha : 3\pi/2 \longrightarrow 2\pi & \Longleftrightarrow \quad \phi = 0 \text{ and } \theta : \pi \longrightarrow \pi/2. \end{cases} \tag{2}$$

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

$$|\varphi\rangle = \cos\alpha|0\rangle + \sin\alpha|1\rangle. \tag{3}$$

A 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

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

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

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

2.2 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 quantum computation.

The definition of a single qubit rotation gate is written as

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

Let the quantum state $|\phi\rangle = \left[\begin{smallmatrix} \cos\theta_0 \\ \sin\theta_0 \end{smallmatrix}\right]$, then $|\phi\rangle$ can be transformed by $R(\theta)$ as follows

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

It is obvious that $R(\theta)$ shifts the phase of $|\phi\rangle$.

2.3 Unitary operators and tensor products

A matrix U is said to be *unitary* if $(U^*)^T U = I$, 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 each of 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 multiparticle 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 two-dimensional 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 $V \otimes V$.

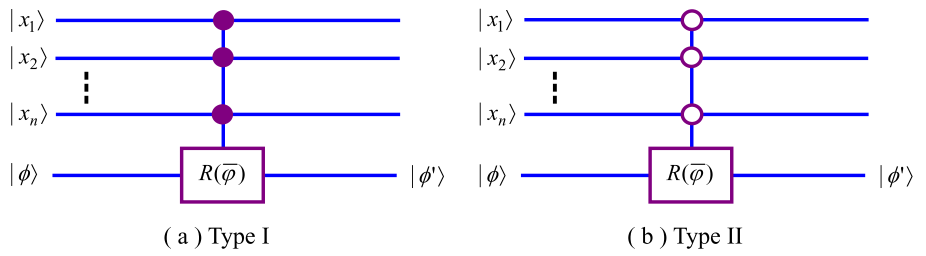


Fig. 2 Multi-qubits controlled-rotation gate

2.4 Multi-qubits controlled-rotation gate

In a true quantum system, a single qubit state is often affected by a joint control of multi-qubits. A multi-qubits controlled-rotation gate $C^n(R)$ is a kind of control model. The multi-qubits system is also described by the wave function $|x_1x_2\cdots x_n\rangle$. In a $(n+1)$-bits quantum system, when the target bit is simultaneously controlled by n input bits, the input/output relationship of the system can be described by multi-qubits controlled-rotation gate in Fig. 2.

In Fig. 2(a), suppose we have $n+1$ qubits, and then we define the controlled operation $C^n(R)$ as follows

$$\begin{aligned} &C^n(R)|x_1x_2\cdots x_n\rangle|\phi\rangle \\ &\quad = |x_1x_2\cdots x_n\rangle R^{x_1x_2\cdots x_n}|\phi\rangle, \end{aligned} \tag{8}$$

where $x_1x_2\cdots x_n$ in the exponent of R means the product of the bits $x_1, x_2, \dots, x_n$. That is, the operator R is applied to last a qubit if the first n qubits are all equal to one; otherwise, nothing is done.

Suppose that the $|x_i\rangle = \cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle$ are the control qubits, and the $|\phi\rangle = \cos(\varphi)|0\rangle + \sin(\varphi)|1\rangle$ is the target qubit. From Eq. (8), the output of $C^n(R)$ is written by equation

$$\begin{aligned} &C^n(R)|x_1x_2\cdots x_n\rangle|\phi\rangle \\ &\quad = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \otimes |\phi\rangle \\ &\qquad + \prod_{i=1}^{n} \sin(\theta_i)\big(\cos(\varphi+\overline{\varphi}) - \cos(\varphi)\big)|\overbrace{11\cdots1}^{n}0\rangle \\ &\qquad + \prod_{i=1}^{n} \sin(\theta_i)\big(\sin(\varphi+\overline{\varphi}) - \sin(\varphi)\big)|\overbrace{11\cdots1}^{n}1\rangle. \end{aligned} \tag{9}$$

We say that a state of a composite system having the property that it can't be written as a product of states of its component systems is an entangled state. For reasons which nobody fully understands, entangled states play a crucial role in quantum computation and quantum information. It is observed from Eq. (9) that the output of $C^n(R)$ is in the entangled state of $n+1$ qubits, and the probability of the target qubit state $|\phi'\rangle$, in which $|1\rangle$ is observed, equals to

$$P = \prod_{i=1}^{n} \sin^2(\theta_i)\big(\sin^2(\varphi+\overline{\varphi}) - \sin^2(\varphi)\big) + \sin^2(\varphi). \tag{10}$$

In Fig. 2(b), the operator R is applied to last a qubit if the first n qubits are all equal to zero, and otherwise, nothing is done. The controlled operation $C^n(R)$ can be defined by the equation

$$C^n(R)|x_1x_2\cdots x_n\rangle|\phi\rangle = |x_1x_2\cdots x_n\rangle R^{\overline{x_1+\cdots+x_n}}|\phi\rangle. \tag{11}$$

By a similar analysis with Fig. 2(a), the probability of the target qubit state $|\phi'\rangle$, in which $|1\rangle$ is observed, equals to

$$P = \prod_{i=1}^{n} \cos^2(\theta_i)\big(\sin^2(\varphi+\overline{\varphi}) - \sin^2(\varphi)\big) + \sin^2(\varphi). \tag{12}$$

At this time, after the joint control of the n input bits, the target bit $|\phi'\rangle$ can be defined as follows

$$|\phi'\rangle = \sqrt{1-P}|0\rangle + \sqrt{P}|1\rangle. \tag{13}$$

3 The QNNSI model

3.1 The quantum-inspired neuron based on controlled-rotation gate

In this section, we first propose a quantum-inspired neuron model based on controlled-rotation gate, as shown in Fig. 3. This model consists of quantum rotation gates and multi-qubits controlled-rotation gate. The $\{|x_i(t_r)\rangle\}$ defined in time domain interval [0, T] denote the input sequences, where $t_r \in [0, \mathrm{T}]$. The $|y\rangle$ denotes the spatial and temporal aggregation results in [0, T]. The output is the probability amplitude of $|1\rangle$ after measuring $|y\rangle$. The control parameters are the rotation angles $\overline{\theta}_i(t_r)$, $\overline{\varphi}(t_r)$, $i = 1, 2, \dots, n$, $r = 1, 2, \dots, q$, n denotes the number of input space dimension, q denotes the length of input sequence.

Unlike classical neuron, each input sample of quantum-inspired neuron is described as a matrix instead of a vector.

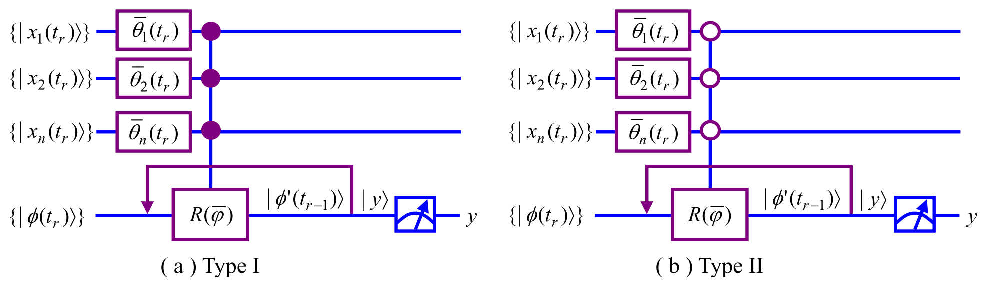


Fig. 3 The model of quantum-inspired neuron based on controlled rotation gate

For example, a single input sample can be written as

$$\begin{bmatrix}\{|x_1(t_r)\rangle\}\\ \{|x_2(t_r)\rangle\}\\ \cdots\\ \{|x_n(t_r)\rangle\}\end{bmatrix}=\begin{bmatrix}|x_1(t_1)\rangle & |x_1(t_2)\rangle & \cdots & |x_1(t_q)\rangle\\ |x_2(t_1)\rangle & |x_2(t_2)\rangle & \cdots & |x_2(t_q)\rangle\\ \cdots & \cdots & \cdots & \cdots\\ |x_n(t_1)\rangle & |x_n(t_2)\rangle & \cdots & |x_n(t_q)\rangle\end{bmatrix}. \tag{14}$$

Suppose $|x_i(t_r)\rangle = \cos\theta_i(t_r)|0\rangle + \sin\theta_i(t_r)|1\rangle$, $|\phi(t_1)\rangle = |0\rangle$. Let

$$\bar h_r = \begin{cases}\prod_{i=1}^n \sin(\theta_i(t_r)+\bar\theta_i(t_r)), & \text{for Fig. 3(a)},\\ \prod_{i=1}^n \cos(\theta_i(t_r)+\bar\theta_i(t_r)), & \text{for Fig. 3(b)}.\end{cases} \tag{15}$$

According to the definition of quantum rotation gate and multi-qubits controlled-rotation gate, the $|\phi'(t_1)\rangle$ is given by

$$|\phi'(t_1)\rangle = \sqrt{1-\left(\bar h_1\sin\bar\varphi(t_1)\right)^2}|0\rangle + \bar h_1\sin\bar\varphi(t_1)|1\rangle. \tag{16}$$

Let $t = t_r$, $r = 2, 3, \dots, q$, from $|\phi(t_r)\rangle = |\phi'(t_{r-1})\rangle$, the aggregate results of quantum neuron in [0,T] is finally written as

$$|y\rangle = |\phi'(t_q)\rangle = \cos\varphi(t_q)|0\rangle + \sin\varphi(t_q)|1\rangle, \tag{17}$$

where $\varphi(t_q) = \arcsin(\{(\bar h_q)^2(\sin^2(\varphi(t_{q-1})+\bar\varphi(t_q)) - \sin^2(\varphi(t_{q-1}))) + \sin^2(\varphi(t_{q-1}))\}^{1/2})$.

In this paper, we define the output of the quantum neuron as the probability amplitude of the corresponding state, in which $|1\rangle$ is observed. Let $h(t_r)$ denote the probability amplitude of the state $|1\rangle$ in $|\phi'(t_r)\rangle$. Using some trigonometry, the output of the quantum neuron is rewritten as

$$y = h(t_q) = \sqrt{(\bar h_q)^2 U_q + \left(h(t_{q-1})\right)^2}, \tag{18}$$

where $U_q = h(t_{q-1})\sqrt{1-(h(t_{q-1}))^2}\sin(2\bar\varphi(t_q)) + (1-2(h(t_{q-1}))^2)\sin^2(\bar\varphi(t_q))$, $h(t_1) = \bar h_1\sin(\bar\varphi(t_1))$.

3.2 The QNNSI model

In this paper, the QNNSI model is shown in Fig. 4, where the hidden layer consists of p quantum-inspired neurons based on controlled-rotation gate (**Type I** is employed for odd serial number, and **type II** is employed for even serial number), $\{|x_1(t_r)\rangle\}, \{|x_2(t_r)\rangle\}, \dots, \{|x_n(t_r)\rangle\}$ denote the input sequences, $h_1, h_2, \dots, h_p$ denote the hidden output, the activation function in hidden layer employs the Eq. (18), the output layer consists of m classical neurons, w_{jk} denote the connection weights in output layer, $y_1, y_2, \dots, y_m$ denote the network output, and the activation function in output layer employs the *Sigmoid* function.

For the lth sample, suppose $|x_i^l(t_r)\rangle = \cos\theta_i^l(t_r)|0\rangle + \sin\theta_i^l(t_r)|1\rangle$, $0 = t_1 < t_2 < \cdots < t_q = \mathrm{T}$ denote the discrete sampling time points, set $|\phi_j^l(t_1)\rangle = |0\rangle$, $j = 1, 2, \dots, p$. Let

$$\bar h_{jr}^l = \begin{cases}\prod_{i=1}^n \sin(\theta_i^l(t_r)+\theta_{ij}(t_r)), & j = 1, 3, 5, \dots,\\ \prod_{i=1}^n \cos(\theta_i^l(t_r)+\theta_{ij}(t_r)), & j = 2, 4, 6, \dots.\end{cases} \tag{19}$$

According to the input/output relationship of quantum neuron, the output of the jth quantum neuron in hidden layer can be written as

$$h_j^l = h_j^l(t_q) = \sqrt{\left(\bar h_{jq}^l\right)^2 U_{jq}^l + \left(h_j^l(t_{q-1})\right)^2}, \tag{20}$$

where $U_{jq}^l = h_j^l(t_{q-1})\sqrt{1-(h_j^l(t_{q-1}))^2}\sin(2\bar\varphi_j(t_q)) + (1-2(h_j^l(t_{q-1}))^2)\sin^2(\bar\varphi_j(t_q))$, $h_j^l(t_1) = \bar h_{j1}^l\sin(\bar\varphi(t_1))$.

The kth output in output layer can be written as

$$y_k^l = \frac{1}{1+e^{-\sum_{j=1}^p w_{jk}h_j^l}}, \tag{21}$$

where $i = 1, 2, \dots, n$, $j = 1, 2, \dots, p$, $k = 1, 2, \dots, m$, $l = 1, 2, \dots, L$, L denotes the total number of samples.

4 The learning algorithm of QNNSI

4.1 The pretreatment of the input and output samples

Set the sampling time points $0 = t_1 < t_2 < \cdots < t_q = \mathrm{T}$. Suppose the lth sample in n-dimensional input space $\{\overline{X}^l(t_r)\} = [\{\bar x_1^l(t_r)\}, \dots, \{\bar x_n^l(t_r)\}]^{\mathrm{T}}$, where $r = 1, 2, \dots, q$,

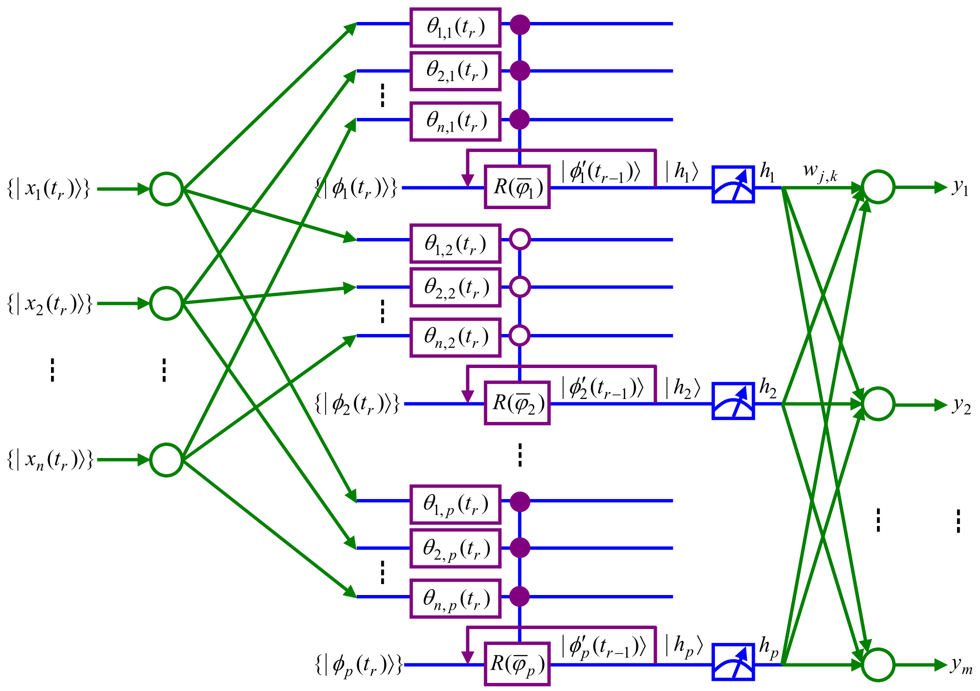


Fig. 4 The model of quantum-inspired neural network with sequence input based on controlled-rotation gate

$l = 1, 2, \ldots, L$. Let

$$\begin{cases} \text{Max}_{i,r} = \max(\overline{x}_i^1(t_r), \overline{x}_i^2(t_r), \ldots, \overline{x}_i^L(t_r)), \\ \text{Min}_{i,r} = \min(\overline{x}_i^1(t_r), \overline{x}_i^2(t_r), \ldots, \overline{x}_i^L(t_r)), \end{cases}$$
$$i = 1, 2, \ldots, n, \tag{22}$$

$$\theta_i^l(t_r) = \begin{cases} \frac{\overline{x}_i^l(t_r) - \text{Min}_{i,r}}{\text{Max}_{i,r} - \text{Min}_{i,r}} \frac{\pi}{2}, & \text{if } \text{Max}_{i,r} > \text{Min}_{i,r}, \\ \frac{\pi}{2}, & \text{if } \text{Max}_{i,r} = \text{Min}_{i,r} \neq 0, \\ 0, & \text{if } \text{Max}_{i,r} = \text{Min}_{i,r} = 0. \end{cases} \tag{23}$$

These samples can be converted into the quantum states as follows

$$\{|X^l(t_r)\rangle\} = [\{|x_1^l(t_r)\rangle\}, \{|x_2^l(t_r)\rangle\}, \ldots, \{|x_n^l(t_r)\rangle\}]^{\text{T}}, \tag{24}$$

where $|x_i^l(t_r)\rangle = \cos(\theta_i^l(t_r))|0\rangle + \sin(\theta_i^l(t_r))|1\rangle$.

It is worth pointing out that although a n-qubit system has 2^n computational basis states, this n-qubit system may form the superpositions of 2^n basis states. Although the number of these superpositions is infinite, in our approach, the superposition can be uniquely determined by the method of converting input samples into quantum states. Hence, the difference between our approach and a single input, zero-hidden layer, and one neuron ANN output, where input $= n$ nodes, is embodied in the following two aspects. (1) For the former, the input sample is a specific quantum superposition state, and for the latter, the input sample is a specific real value vector. (2) For the former, the activation functions are designed through quantum computing principle, and for the latter, the classical *Sigmoid* functions are used as the activation functions.

Similarly, suppose the lth output sample $\{\overline{Y}^l\} = [\{\overline{y}_1^l\}, \{\overline{y}_2^l\}, \ldots, \{\overline{y}_m^l\}]^{\text{T}}$, where $l = 1, 2, \ldots, L$. Let

$$\begin{cases} \text{Max}_k = \max(\overline{y}_k^1, \overline{y}_k^2, \ldots, \overline{y}_k^L), \\ \text{Min}_k = \min(\overline{y}_k^1, \overline{y}_k^2, \ldots, \overline{y}_k^L), \end{cases} \tag{25}$$

then, these output samples can be normalized by the following equation

$$\overline{y}_k^l = \begin{cases} \frac{\overline{y}_k^l - \text{Min}_k}{\text{Max}_k - \text{Min}_k}, & \text{if } \text{Max}_k > \text{Min}_k, \\ 1, & \text{if } \text{Max}_k = \text{Min}_k \neq 0, \\ 0, & \text{if } \text{Max}_k = \text{Min}_k = 0, \end{cases} \tag{26}$$

where $k = 1, 2, \ldots, m$.

4.2 The adjustment of QNNSI parameters

The adjustable parameters of QNNSI include: (1) the rotation angles of quantum rotation gates in hidden layer: $\theta_{ij}(t_r)$ and $\overline{\varphi}_j(t_r)$; (2) the connection weights in output layer: w_{jk}.

Because the number of parameters is greater and gradient calculation is more complicated, the standard gradient descent algorithm is not easy to converge. Hence we employ the *Levenberg–Marquardt* algorithm in Ref. [22] to adjust the QNNSI parameters. Suppose $\overline{y}_1^l, \overline{y}_2^l, \ldots, \overline{y}_m^l$ denote the normalized desired outputs of the lth sample, and $y_1^l, y_2^l, \ldots, y_m^l$ denote the corresponding actual outputs. The evaluation function is defined as follows

$$E=\max_{1\le l\le L}\max_{1\le k\le m}\left|e_k^l\right|=\max_{1\le l\le L}\max_{1\le k\le m}\left|\overline{y}_k^l-y_k^l\right|. \qquad (27)$$

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

$$\mathbf{p}^{\mathrm{T}}=\left[\theta_{1,1}(t_1),\ldots,\theta_{n,p}(t_q),\overline{\varphi}_1(t_1),\ldots,\overline{\varphi}_p(t_q),\right.$$
$$\left.w_{1,1},\ldots,w_{pm}\right], \qquad (28)$$

$$\mathbf{e}^{\mathrm{T}}(\mathbf{p})=\left[e_1^1,e_2^1,\ldots,e_m^1,\ e_1^2,e_2^2,\ldots,e_m^2,\ldots,\right.$$
$$\left.e_1^L,e_2^L,\ldots,e_m^L\right], \qquad (29)$$

$$\mathbf{J}(\mathbf{p})=\begin{bmatrix}
\frac{\partial e_1^1}{\partial\theta_{1,1}(t_1)} & \cdots & \frac{\partial e_1^1}{\partial\theta_{n,p}(t_q)} & \frac{\partial e_1^1}{\partial\overline{\varphi}_1(t_1)} & \cdots & \frac{\partial e_1^1}{\partial\overline{\varphi}_p(t_q)} & \frac{\partial e_1^1}{\partial w_{1,1}} & \cdots & \frac{\partial e_1^1}{\partial w_{p,m}}\\
\vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots\\
\frac{\partial e_m^1}{\partial\theta_{1,1}(t_1)} & \cdots & \frac{\partial e_m^1}{\partial\theta_{n,p}(t_q)} & \frac{\partial e_m^1}{\partial\overline{\varphi}_1(t_1)} & \cdots & \frac{\partial e_m^1}{\partial\overline{\varphi}_p(t_q)} & \frac{\partial e_m^1}{\partial w_{1,1}} & \cdots & \frac{\partial e_m^1}{\partial w_{p,m}}\\
\frac{\partial e_1^2}{\partial\theta_{1,1}(t_1)} & \cdots & \frac{\partial e_1^2}{\partial\theta_{n,p}(t_q)} & \frac{\partial e_1^2}{\partial\overline{\varphi}_1(t_1)} & \cdots & \frac{\partial e_1^2}{\partial\overline{\varphi}_p(t_q)} & \frac{\partial e_1^2}{\partial w_{1,1}} & \cdots & \frac{\partial e_1^2}{\partial w_{p,m}}\\
\vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots\\
\frac{\partial e_m^2}{\partial\theta_{1,1}(t_1)} & \cdots & \frac{\partial e_m^2}{\partial\theta_{n,p}(t_q)} & \frac{\partial e_m^2}{\partial\overline{\varphi}_1(t_1)} & \cdots & \frac{\partial e_m^2}{\partial\overline{\varphi}_p(t_q)} & \frac{\partial e_m^2}{\partial w_{1,1}} & \cdots & \frac{\partial e_m^2}{\partial w_{p,m}}\\
\vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots\\
\frac{\partial e_1^L}{\partial\theta_{1,1}(t_1)} & \cdots & \frac{\partial e_1^L}{\partial\theta_{n,p}(t_q)} & \frac{\partial e_1^L}{\partial\overline{\varphi}_1(t_1)} & \cdots & \frac{\partial e_1^L}{\partial\overline{\varphi}_p(t_q)} & \frac{\partial e_1^L}{\partial w_{1,1}} & \cdots & \frac{\partial e_1^L}{\partial w_{p,m}}\\
\vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots\\
\frac{\partial e_m^L}{\partial\theta_{1,1}(t_1)} & \cdots & \frac{\partial e_m^L}{\partial\theta_{n,p}(t_q)} & \frac{\partial e_m^L}{\partial\overline{\varphi}_1(t_1)} & \cdots & \frac{\partial e_m^L}{\partial\overline{\varphi}_p(t_q)} & \frac{\partial e_m^L}{\partial w_{1,1}} & \cdots & \frac{\partial e_m^L}{\partial w_{p,m}}
\end{bmatrix}, \qquad (30)$$

where the gradient calculations in $\mathbf{J}(\mathbf{p})$ see to the Appendix.

According to *Levenberg–Marquardt* algorithm, the iterative equation of adjusting QNNSI parameters is written as follows

$$\mathbf{p}_{t+1}=\mathbf{p}_t-\left(\mathbf{J}^{\mathrm{T}}(\mathbf{p}_t)\mathbf{J}(\mathbf{p}_t)+\mu_t\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}}(\mathbf{p}_t)\mathbf{e}(\mathbf{p}_t), \qquad (31)$$

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

4.3 The stopping criterion of QNNSI

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.

4.4 Learning algorithm description

The structure of QNNSI is shown in the following.

```
Procedure QNNSI
Begin
    $t \leftarrow 0$
(1) The pretreatment of the input and output samples.
(2) Initialization of QNNSI, including
    (a) the predefined precision $\varepsilon$,
    (b) the predefined maximum number of iterative steps $N$,
    (c) the parameter of Levenberg–Marquardt algorithm $\mu_t$,
    (d) the parameters of QNNSI $\{\theta_{ij}(t_r),\overline{\varphi}_j(t_r)\}\in(-\frac{\pi}{2},\frac{\pi}{2})$, $\{w_{jk}\}\in(-1,1)$.
(3) While (not termination-condition)
    Begin
        (a) computing the actual outputs of all samples by Eqs. (19)–(21),
        (b) computing the value of the evaluation function $E$ by Eq. (27),
        (c) adjusting the parameters $\{\theta_{ij}(t_r)\}$, $\{\overline{\varphi}_j(t_r)\}$, $\{w_{jk}\}$ by Eq. (31).
        (d) $t \leftarrow t+1$,
    End
End
```

4.5 Diagnostic explanatory capabilities

Finally, we briefly give the diagnostic explanatory capabilities of QNNSI, namely, given the complex model, how can one explain a given prediction, inference, or classification based on QNNSI. We believe that any given prediction, inference, or classification can be seen as an approximation problem from the input space to the output space. In this sense, the above problem is converted into the design problem of multi-dimension sequence samples. Our approach is below. For a n-dimension sample X of classical ANN, if n is a prime number, then extend the dimensions of this sample X to $m = n + 1$ by setting $X(m)$ equal $X(n)$, and otherwise, nothing is done. We decompose m into the product of m_1 and m_2 and make these two numbers as close as possible. At this time, a n-dimension sample X of ANN is converted into a m_1 dimension sequence sample of QNNSI where the sequence length equals m_2, or a m_2 dimension sequence sample of QNNSI where the sequence length equals m_1.

5 Simulations

In order to experimentally illustrate the effectiveness of the proposed QNNSI, four examples are used to compare it with the ANN with a hidden layer in this section. In these experiments, we perform and evaluate the QNNSI in Matlab (Version 7.1.0.246) on a Windows PC with 2.19 GHz CPU and 1.00 GB RAM. Our QNNSI has the same structure and parameters as the ANN in these experiments, and the same *Levenberg–Marquardt* algorithm in Ref. [22] is applied in two models. Some relevant concepts are defined as follows.

Approximation error Suppose $[\overline{y}_1^l, \overline{y}_2^l, \dots, \overline{y}_m^l]$ and $[y_1^l, y_2^l, \dots, y_m^l]$ denote the lth desired output and the corresponding actual output after training, respectively. The approximation error is defined as

$$E = \max_{1\le l\le L} \max_{1\le k\le m} \left|\overline{y}_k^l - y_k^l\right|, \tag{32}$$

where L denotes the number of the training samples, and m denotes the dimension of the output space.

Average approximation error Suppose $E_1, E_2, \dots, E_N$ denote the approximation error over N training trials, respectively. The average approximation error is defined as

$$E_{avg} = \frac{1}{N}\sum_{i=1}^{N} E_i. \tag{33}$$

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

$$\lambda = \frac{C}{N}. \tag{34}$$

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

Average iterative steps Suppose $S_1, S_2, \dots, S_N$ denote the iterative steps over N training trials, respectively. The average iterative steps are defined as

$$S_{avg} = \frac{1}{N}\sum_{i=1}^{N} S_i. \tag{35}$$

Average running time Suppose $T_1, T_2, \dots, T_N$ denote the running time over N training trials, respectively. The average running time is defined as

$$T_{avg} = \frac{1}{N}\sum_{i=1}^{N} T_i. \tag{36}$$

5.1 Time series prediction for Mackey–Glass

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

$$x(t+1) - x(t) = a\frac{x(t-\tau)}{1+x^{10}(t-\tau)} - bx(t), \tag{37}$$

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 schemes is to employ n data adjacent to each other to predict the next one data. Namely, in our model, the sequence length equals to n. Therefore, each sample consists of n input values and an output value. Hence, there is only one output node in QNNSI and ANN. In order to fully compare the approximation ability of two models, the number of hidden nodes are respectively set to $10, 11, \dots, 30$. The predefined precision is set to 0.05, and the maximum of iterative steps is set to 100. The QNNSI rotation angles in hidden layer are initialized to random numbers in $(-\pi/2, \pi/2)$, and the connection weights in output layer are initialized to random numbers in $(-1, 1)$. For ANN, all weights are initialized to random numbers in $(-1, 1)$, and the *Sigmoid* functions are used as activation functions in hidden layer and output layer.

Obviously, ANN has n input nodes, and an ANN's input sample can be described as a n-dimensional vector. For the number of input nodes of QNNSI, we employ the following six kinds of settings shown in Table 1. For each of these settings in Table 1, a single QNNSI input sample can be described as a matrix.

It is worth noting that, in QNNSI, a $n \times q$ matrix can be used to describe a single sequence sample. In general, ANN cannot deal directly with a single $n \times q$ sequence sample. In ANN, a $n \times q$ matrix is usually regarded as q n-dimensional vector samples. For fair comparison, in ANN, we have expressed the $n \times q$ sequence samples into the nq-dimensional vector samples. Therefore, in Table 1, the sequence lengths for ANN are not changed. It is clear that, in fact, there is only one kind of ANN in Table 1, namely, ANN32.

Our experiment scheme is that, for each kind of combination of input nodes and hidden nodes, six QNNSIs and one ANN 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 QNNSI with ANN. Training results contrast are shown in Tables 2, 3, 4 and 5, where QNNSIn_q denotes QNNSI with n input nodes and q sequence length.

From Tables 2–5, we can see that when the input nodes take 4 and 8, the performance of QNNSIs are obviously superior to that of ANN, and the QNNSIs have better stability than ANN when the number of hidden nodes changes. The same results also are illustrated in Figs. 5, 6, 7 and 8.

Next, we investigate the generalization ability of QNNSI. Based on the above experimental results, we only investigate QNNSI4_8 and QNNSI8_4. Our experiment scheme is that two QNNSIs and one ANN train 10 times on the training set, and the generalization ability is immediately investigated on the testing set after each training. The average results of the 10 tests are regarded as the evaluation indexes. We first present the following definition of evaluation indexes.

Average prediction error Suppose $[\overline{y}_1^l, \overline{y}_2^l, \ldots, \overline{y}_m^l]$ and $[\widehat{y}_1^l(t), \widehat{y}_2^l(t), \ldots, \widehat{y}_m^l(t)]$ denote the desired output of the lth sample and the corresponding prediction output after the tth testing respectively. The average prediction error over N testing is defined as

$$\overline{E}_{avg} = \frac{1}{N}\sum_{t=1}^{N} \max_{1\le l\le L} \max_{1\le k\le m} \left|\overline{y}_k^l - \widehat{y}_k^l(t)\right|, \tag{38}$$

Table 1 The input nodes and the sequence length setting of QNNSIs and ANN

QNNSIs		ANN	
Input nodes	Sequence length	Input nodes	Sequence length
1	32	32	1
2	16	32	1
4	8	32	1
8	4	32	1
16	2	32	1
32	1	32	1

Table 2 Training results of average approximation error

Model	Hidden nodes										
	10	12	14	16	18	20	22	24	26	28	30
QNNSI1_32	0.0430	0.0400	0.2275	0.0412	0.1321	0.1317	0.1344	0.3155	0.4100	0.2242	0.4101
QNNSI2_16	0.0455	0.0432	0.1326	0.1344	0.0426	0.2266	0.2252	0.1348	0.1349	0.1341	0.1351
QNNSI4_8	**0.0426**	**0.0417**	**0.0420**	**0.0430**	**0.0419**	**0.0434**	**0.0418**	**0.0417**	**0.0428**	**0.0431**	**0.0425**
QNNSI8_4	**0.0433**	**0.0427**	**0.0437**	**0.0428**	**0.0431**	**0.0427**	**0.0426**	**0.0444**	**0.0431**	**0.0439**	**0.0441**
QNNSI16_2	0.0859	0.0469	0.0819	0.0435	0.0445	0.0440	0.0446	0.0429	0.0424	0.0430	0.0431
QNNSI32_1	0.4746	0.4770	0.4746	0.4746	0.4741	0.4744	0.4746	0.4779	0.4743	0.4746	0.4745
ANN32	0.3198	0.2295	0.1377	0.1367	0.2292	0.0453	0.4122	0.1377	0.2284	0.1371	0.2287

Table 3 Training results of average iterative steps

Model	Hidden nodes										
	10	12	14	16	18	20	22	24	26	28	30
QNNSI1_32	10.0	9.90	27.3	7.90	16.5	16.9	16.8	34.2	43.8	25.1	43.6
QNNSI2_16	7.50	6.50	15.7	15.1	5.70	24.5	24.1	14.8	14.5	14.1	14.7
QNNSI4_8	**6.10**	**5.40**	**6.00**	**5.10**	**6.10**	**4.70**	**4.90**	**4.70**	**4.20**	**4.80**	**4.60**
QNNSI8_4	**6.90**	**6.30**	**6.10**	**5.80**	**5.80**	**6.00**	**5.30**	**5.60**	**5.00**	**5.10**	**4.90**
QNNSI16_2	34.1	40.5	31.8	15.1	12.3	11.6	10.3	9.90	9.90	10.7	8.40
QNNSI32_1	100	100	100	100	100	100	100	100	100	100	100
ANN32	45.8	32.2	23.9	23.1	33.4	14.8	47.6	21.1	29.8	21.2	30.1

Table 4 Training results of average running time (s)

Model	Hidden nodes										
	10	12	14	16	18	20	22	24	26	28	30
QNNSI1_32	80.30	100.12	306.59	99.49	253.26	301.29	348.72	656.19	913.15	584.80	1061.1
QNNSI2_16	29.93	34.94	93.58	109.2	51.36	230.8	241.2	169.3	171.0	165.3	199.1
QNNSI4_8	**17.65**	**16.55**	**20.21**	**20.72**	**29.15**	**29.67**	**30.04**	**33.41**	**36.48**	**42.67**	**51.44**
QNNSI8_4	**12.45**	**14.43**	**17.29**	**21.44**	**26.27**	**30.56**	**32.73**	**38.86**	**39.17**	**41.23**	**45.37**
QNNSI16_2	54.66	65.36	63.51	38.36	34.07	39.18	39.91	46.62	48.04	56.55	46.07
QNNSI32_1	122.22	132.80	150.90	174.93	221.64	237.06	250.45	287.07	386.51	439.71	428.34
ANN32	23.02	22.33	21.73	26.01	42.95	26.42	83.37	47.70	74.48	63.87	99.36

Table 5 Training results of convergence ratio (%)

Model	Hidden nodes										
	10	12	14	16	18	20	22	24	26	28	30
QNNSI1_32	100	100	80	100	90	90	90	70	60	80	60
QNNSI2_16	100	100	90	90	100	80	80	90	90	90	90
QNNSI4_8	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
QNNSI8_4	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
QNNSI16_2	90	90	90	100	100	100	100	100	100	100	100
QNNSI32_1	0	0	0	0	0	0	0	0	0	0	0
ANN32	70	80	90	90	80	90	60	90	80	90	80

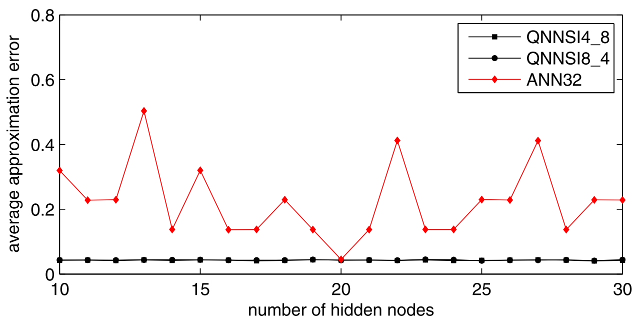


Fig. 5 The average approximation error contrast

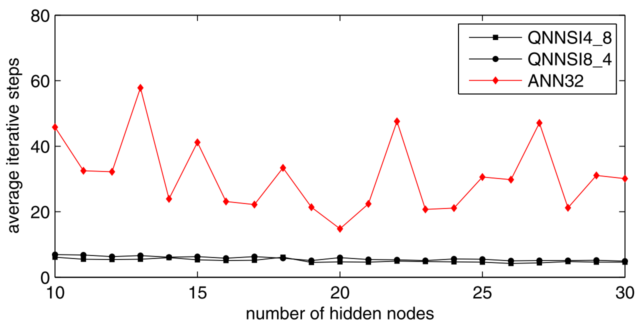


Fig. 6 The average iterative steps contrast

where m denotes the dimension of the output space, L denotes the number of the testing samples.

Average error mean Suppose $\overline{y}^l = [\overline{y}_1^l, \overline{y}_2^l, \dots, \overline{y}_m^l]$ and $\widehat{y}^l(t) = [\widehat{y}_1^l(t), \widehat{y}_2^l(t), \dots, \widehat{y}_m^l(t)]$ denote the desired output of the lth sample and the corresponding prediction output after the tth testing respectively. The average error mean over N testing is defined as

$$\overline{E}_{mean} = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left|\overline{y}^l - \widehat{y}^l(t)\right|, \tag{39}$$

Average prediction variance Suppose $\overline{y}^l = [\overline{y}_1^l, \overline{y}_2^l, \dots, \overline{y}_m^l]$ and $\widehat{y}^l(t)= [\widehat{y}_1^l(t), \widehat{y}_2^l(t), \dots, \widehat{y}_m^l(t)]$ denote the desired output of the lth sample and the corresponding prediction output after the tth testing respectively. The average error variance over N testing is defined as

$$\overline{E}_{var} = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{L-1}\sum_{l=1}^{L}\left(\left|\overline{y}^l - \widehat{y}^l(t)\right| - \frac{1}{L}\sum_{l=1}^{L}\left|\overline{y}^l - \widehat{y}^l(t)\right|\right)^2, \tag{40}$$

The evaluation indexes contrast of QNNSIs and ANN are shown in Table 6. Taking 24 hidden nodes for example, and the average prediction results contrast over 10 testing are illustrated in Fig. 9. The experimental results show that the generalization ability of the two QNNSIs is obviously superior to that of ANN.

These experimental results can be explain as follows. For processing of input information, QNNSI and ANN take two different approaches. QNNSI directly receives a discrete input sequence. In QNNSI, using quantum information processing mechanism, the input is circularly mapped to the output of quantum controlled-rotation gates in hidden layer. As the controlled-rotation gate's output is in the entangled state of multi-qubits, therefore, this mapping is highly nonlinear, which makes QNNSI have the stronger approximation ability. In addition, QNNSI's each input sample can be described as a matrix with n rows and q columns. It is clear from QNNSI's algorithm that, for the different combination of n and q, the output of quantum-inspired neuron in hidden layer is also different. In fact, The number of discrete points q denotes the *depth* of pattern memory, and the number of input nodes n denotes the *breadth* of pattern memory. When the *depth* and the *breadth* are appropriately matched, the QNNSI shows excellent performance. For the ANN, because its input can only be described as a nq-dimensional vector, it does not directly deal with a discrete input sequence. Namely, it can only obtain the sample characteristics by way of *breadth* instead of *depth*. Hence, in the ANN information processing, there inevitably exists the loss of sample characteristics, which affects its approximation and generalization ability.

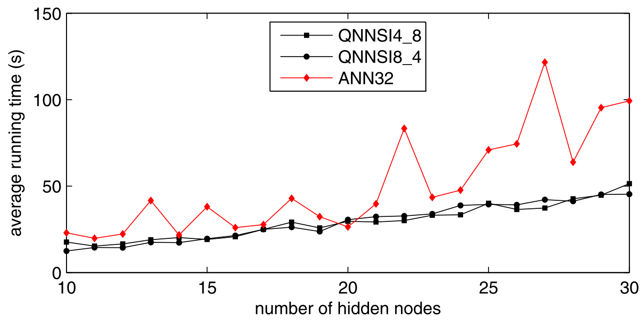


Fig. 7 The average running time contrast

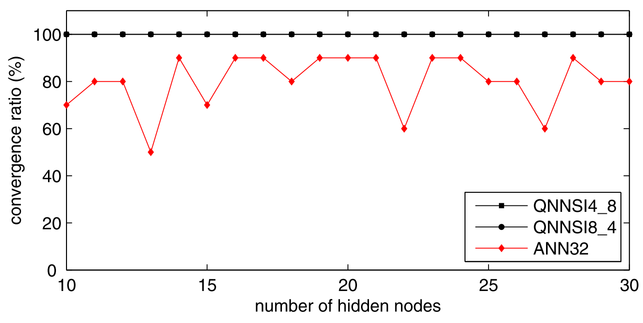


Fig. 8 The convergence ratio contrast

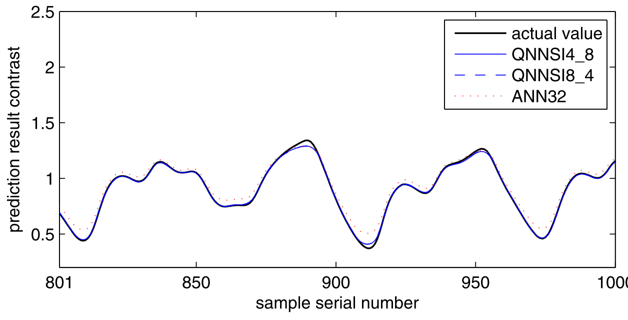


Fig. 9 The average prediction results contrast of QNNSIs and ANN

Table 6 The average prediction error contrast of QNNSIs and ANN

Model	Index	Hidden nodes										
		10	12	14	16	18	20	22	24	26	28	30
QNNSI4_8	$\overline{E}_{avg}$	0.051	0.051	0.050	0.052	0.051	0.054	0.051	0.052	0.053	0.053	0.052
	$\overline{E}_{mean}$	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.010	0.009
	$\overline{E}_{var}$	9e-5	9e-5	9e-5	9e-5	9e-5	1e-4	1e-4	9e-5	1e-4	1e-4	9e-5
QNNSI8_4	$\overline{E}_{avg}$	0.052	0.053	0.052	0.052	0.052	0.052	0.052	0.054	0.052	0.052	0.053
	$\overline{E}_{mean}$	0.009	0.009	0.009	0.008	0.009	0.008	0.008	0.009	0.009	0.009	0.009
	$\overline{E}_{var}$	9e-5	9e-5	9e-5	9e-5	9e-5	9e-5	9e-5	9e-5	9e-5	9e-5	9e-5
ANN32	$\overline{E}_{avg}$	0.332	0.243	0.152	0.153	0.244	0.061	0.425	0.154	0.243	0.153	0.243
	$\overline{E}_{mean}$	0.148	0.105	0.053	0.062	0.107	0.011	0.182	0.053	0.106	0.052	0.095
	$\overline{E}_{var}$	0.019	0.013	0.006	0.006	0.013	0.0001	0.025	0.006	0.013	0.006	0.013

5.2 Annual average of sunspot prediction

In this section, we take the measured data of annual average of sunspot from 1749 to December 2007 as the experiment objects, and investigate the prediction ability of the proposed model. All samples data are shown in Fig. 10. In all samples, we use the first 200 years (1949–1948) data to train the network, and the remaining 59 years (1949–2007) data to test the generalization of the proposed model. For the input nodes and the sequence length, we employ the seven kinds of settings shown in Table 7. In this experiment, we set the number of hidden nodes to 20, 21, ..., 40, respectively. The target error is set to 0.05, and the maximum number of iterative steps is set to 100. The other parameters of QNNSIs and ANNs are set by the same way as the previous experiment.

7 QNNSIs and 2 ANNs are run 10 times respectively for each setting of hidden nodes, and then we use the same evaluation indicators as the previous experiment to compare QNNSIs with ANNs. Training result contrasts are shown in Tables 8, 9, 10 and 11.

From Tables 8–11, we can see that the performance of QNNSI5_10, QNNSI7_7 and QNNSI10_5 are obviously superior to that of the two ANNs. The convergence ratio of these three QNNSIs reaches 100% under a variety of values of hidden nodes. Overall, the other three indicators of these three QNNSIs are better than that of two ANNs, and there is good stability when the number of hidden nodes changes. The same results also are illustrated in Figs. 11, 12, 13 and 14.

Next, we investigate the generalization ability of QNNSI. Based on the above experimental results, we only investigate QNNSI5_10, QNNSI7_7, and QNNSI10_5. Our experiment scheme is that three QNNSIs and two ANNs are respectively done 10 training by the first 200 years (1749–1948) data, and are immediately tested by the remaining 59 years (1949–2007) data after each training. The average prediction error of the 10 tests is regarded as the evaluation index. The average prediction error contrast of QNNSIs and ANNs are shown in Table 12. Taking 35 hidden nodes for example, the average prediction result contrast are illustrated in Fig. 15. The experimental results show that the generalization ability of three QNNSIs is obviously superior to that of corresponding ANNs.

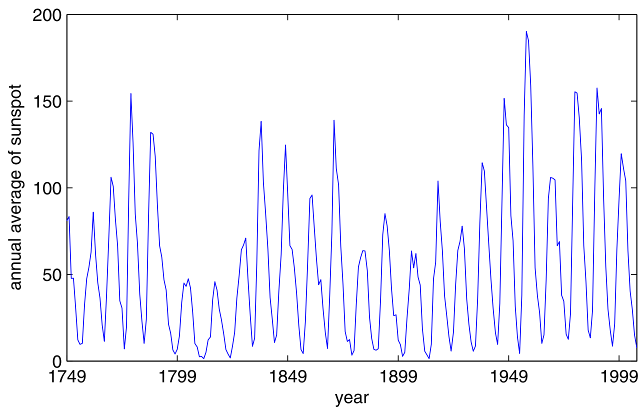


Fig. 10 The measured data of annual average of sunspot

Table 7 The input nodes and the sequence length setting of QNNSIs and ANNs

QNNSIs		ANNs	
Input nodes	Sequence length	Input nodes	Sequence length
1	50	50	1
2	25	50	1
5	10	50	1
7	7	49	1
10	5	50	1
25	2	50	1
50	1	50	1

Table 8 Training results of average approximation error

Model	Hidden nodes										
	20	22	24	26	28	30	32	34	36	38	40
QNNSI1_50	40.20	40.20	38.60	36.60	28.90	20.30	36.90	34.90	52.80	18.00	43.40
QNNSI2_25	19.30	19.40	28.10	36.60	18.40	26.80	36.30	36.00	26.80	17.30	35.70
QNNSI5_10	**12.40**	**12.20**	**10.30**	**10.60**	**9.600**	**10.30**	**11.00**	**9.700**	**10.00**	**9.400**	**8.500**
QNNSI7_7	**15.50**	**13.50**	**13.60**	**13.20**	**12.10**	**12.30**	**12.60**	**11.90**	**11.80**	**10.30**	**10.40**
QNNSI10_5	**18.00**	**14.80**	**15.70**	**14.50**	**15.60**	**13.80**	**14.60**	**13.40**	**14.10**	**12.80**	**13.40**
QNNSI25_2	99.20	85.00	93.30	89.60	82.70	72.20	77.50	60.90	75.20	66.10	71.20
QNNSI50_1	100	100	100	100	100	100	100	100	100	100	100
ANN49	87.90	88.40	84.80	86.20	89.60	85.80	88.00	85.30	82.50	69.90	87.60
ANN50	79.10	92.60	93.20	79.40	92.30	87.60	85.80	86.60	92.40	76.00	87.30

Table 9 Training results of average iterative steps

Model	Hidden nodes										
	20	22	24	26	28	30	32	34	36	38	40
QNNSI1_50	40.69	36.69	40.47	40.40	21.84	8.450	40.54	36.26	72.73	8.144	54.84
QNNSI2_25	26.38	22.84	37.13	51.01	22.57	40.84	50.82	58.47	36.96	22.61	62.47
QNNSI5_10	**9.102**	**9.103**	**9.233**	**8.988**	**9.027**	**9.115**	**9.023**	**8.961**	**9.072**	**8.942**	**9.028**
QNNSI7_7	**9.125**	**9.174**	**9.072**	**9.211**	**9.028**	**8.940**	**9.174**	**8.936**	**9.110**	**8.961**	**9.081**
QNNSI10_5	**9.277**	**8.901**	**9.150**	**9.134**	**9.123**	**9.129**	**9.161**	**9.155**	**8.984**	**9.199**	**9.100**
QNNSI25_2	38.02	39.66	26.08	20.55	22.83	10.97	11.88	9.875	14.43	9.276	9.636
QNNSI50_1	94.40	94.40	94.40	94.40	94.39	94.39	94.38	94.20	94.40	94.40	94.40
ANN49	10.37	10.87	10.15	10.38	10.10	9.828	10.22	9.807	9.765	9.414	9.812
ANN50	9.808	10.38	10.37	9.559	10.46	10.00	9.737	9.708	9.628	9.524	9.896

Table 10 Training results of average running time (s)

Model	Hidden nodes										
	20	22	24	26	28	30	32	34	36	38	40
QNNSI1_50	389.30	489.90	586.80	684.20	665.30	554.70	1191.9	1344.2	2387.4	972.20	2670.7
QNNSI2_25	162.28	221.41	344.19	495.03	250.20	384.93	768.91	837.37	691.63	336.29	707.80
QNNSI5_10	**113.61**	**105.38**	**92.445**	**104.11**	**114.82**	**150.21**	**150.02**	**137.65**	**149.81**	**143.35**	**152.54**
QNNSI7_7	**66.266**	**55.433**	**74.901**	**93.984**	**95.672**	**101.76**	**129.86**	**126.66**	**136.12**	**139.53**	**140.45**
QNNSI10_5	**92.259**	**72.502**	**103.34**	**108.26**	**129.73**	**115.09**	**158.92**	**211.05**	**246.20**	**212.38**	**215.31**
QNNSI25_2	567.20	484.80	478.20	634.80	588.40	419.10	551.80	358.50	492.20	496.80	1283.5
QNNSI50_1	497.20	497.10	634.80	784.10	671.30	789.90	699.60	645.70	627.80	741.10	2659.4
ANN49	314.85	296.63	335.97	407.13	404.54	546.14	677.46	591.49	714.95	613.73	786.97
ANN50	106.07	156.89	197.15	207.75	293.74	334.77	390.02	465.37	578.43	553.22	742.02

Table 11 Training results of convergence ratio (%)

Model	Hidden nodes										
	20	22	24	26	28	30	32	34	36	38	40
QNNSI1_50	80	80	80	80	90	100	80	80	60	100	70
QNNSI2_25	90	90	80	70	90	80	70	70	80	90	70
QNNSI5_10	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
QNNSI7_7	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
QNNSI10_5	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
QNNSI25_2	10	50	30	40	50	70	80	90	70	90	80
QNNSI50_1	0	0	0	0	0	0	0	0	0	0	0
ANN49	40	40	50	40	40	60	50	50	60	80	60
ANN50	80	20	30	70	30	40	60	50	60	80	50

5.3 Caravan insurance policy prediction

In this experiment, we predict who would be interested in buying a caravan insurance policy. This data set used in the CoIL 2000 Challenge contains information on customers of an insurance company and comes from the following url: http://kdd.ics.uci.edu/databases/tic/tic.html. The data was supplied by the Dutch data mining company Sentient Machine Research and is based on a real world business problem. The training set contains 5822 descriptions of customers. Each record consists of 86 attributes, containing sociodemographic data (attribute 1–43) and product ownership (attributes 44–86). The sociodemographic data is derived from zip codes. All customers living in areas with the same zip code have the same sociodemographic attributes. Attribute 86 is the target variable, which equals 0 or 1 and indicate information of whether or not they have a caravan insurance policy. The Dataset for predictions contains 4000

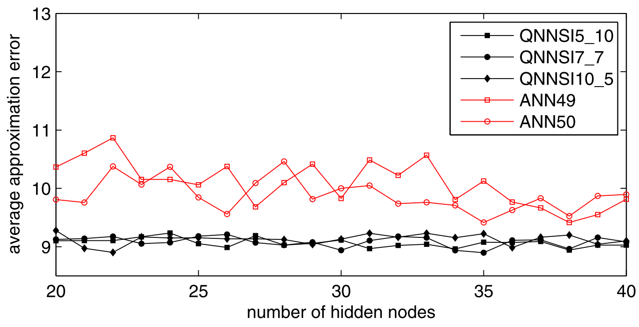


Fig. 11 The average approximation error contrast

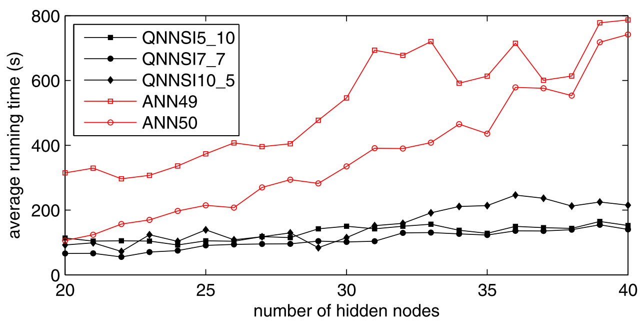


Fig. 13 The average running time contrast

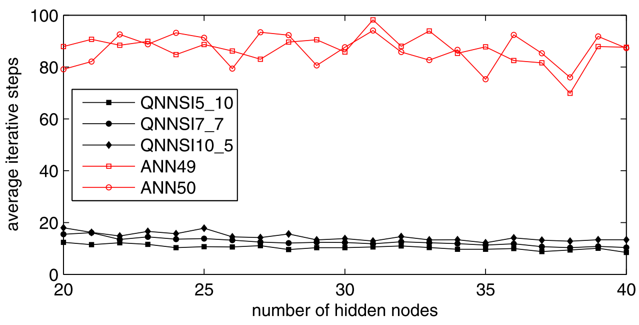


Fig. 12 The average iterative steps contrast

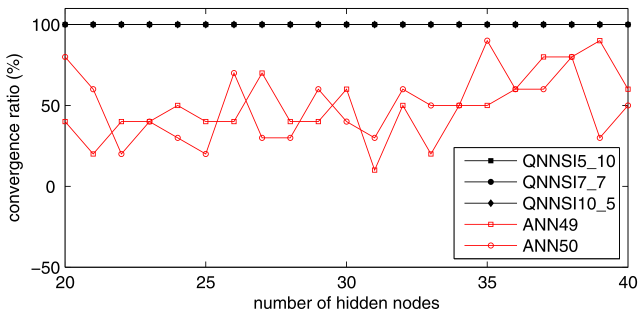


Fig. 14 The convergence ratio contrast

Table 12 The average prediction error contrast of QNNSIs and ANNs

Model	Index	Hidden nodes										
		20	22	24	26	28	30	32	34	36	38	40
QNNSI5_10	$\overline{E}_{avg}$	15.48	15.42	15.17	15.51	16.75	15.44	14.94	16.37	16.16	14.93	14.53
	$\overline{E}_{mean}$	3.506	2.944	3.525	3.777	3.797	3.158	2.994	3.354	3.757	3.078	3.218
	$\overline{E}_{var}$	10.46	8.383	10.30	10.07	11.71	9.256	8.341	10.00	11.15	8.062	8.512
QNNSI7_7	$\overline{E}_{avg}$	15.47	16.19	15.32	17.04	15.32	15.21	15.07	15.49	15.39	14.96	15.52
	$\overline{E}_{mean}$	3.240	3.259	3.032	3.575	3.194	3.206	2.966	3.014	2.934	3.089	3.238
	$\overline{E}_{var}$	9.406	9.440	8.179	10.50	9.464	8.989	8.418	8.593	8.465	8.958	9.592
QNNSI10_5	$\overline{E}_{avg}$	15.99	15.93	15.79	16.11	16.46	15.70	16.20	15.89	15.58	16.45	16.47
	$\overline{E}_{mean}$	4.086	3.788	3.518	3.654	3.943	3.615	3.643	3.727	3.880	3.605	3.760
	$\overline{E}_{var}$	12.40	10.57	10.17	10.81	11.45	10.30	10.04	10.52	10.75	10.85	10.29
ANN49	$\overline{E}_{avg}$	31.40	26.33	27.80	29.07	27.20	61.01	28.26	29.97	45.47	45.19	45.23
	$\overline{E}_{mean}$	10.08	8.159	8.383	9.518	8.360	25.91	8.686	9.294	16.18	15.11	15.54
	$\overline{E}_{var}$	55.32	34.50	37.60	46.60	40.58	578.8	40.18	48.02	311.6	312.8	312.8
ANN50	$\overline{E}_{avg}$	44.20	31.05	28.92	29.49	30.32	30.23	28.56	42.48	57.92	27.67	26.18
	$\overline{E}_{mean}$	14.85	9.221	8.605	8.439	9.534	9.119	8.384	14.77	25.16	9.121	8.356
	$\overline{E}_{var}$	306.2	47.31	44.02	39.37	46.98	49.47	41.74	307.4	569.5	44.21	37.57

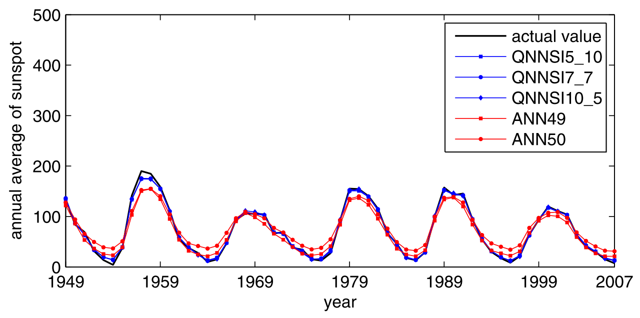


Fig. 15 The average prediction results contrast of QNNSIs and ANNs

Table 13 The input nodes and the sequence length setting of QNNSIs and ANN

QNNSIs		ANN	
Input nodes	Sequence length	Input nodes	Sequence length
1	85	85	1
5	17	85	1
17	5	85	1
85	1	85	1

customer records of whom only the organisers know if they have a caravan insurance policy. It has the same format as the training set, only the target is missing. Participants are supposed to return the list of predicted targets only.

Considering the each customer consists of 85 feature attributes for the input nodes and the sequence length, we employ the four kinds of settings shown in Table 13. In this experiment, we set the number of hidden nodes to $10, 11, \ldots, 20$, respectively. The maximum number of iterative steps is set to 100. The other parameters of QNNSIs and ANN are set by the same way as the previous experiment. In this experiment, we do not set the value of target error. The algorithm is not stopped until it reaches the predefined maximum number of iterative steps.

QNNSIs and ANN train 10 times for each setting of hidden nodes by the training set data, and are immediately tested by the testing set data after each training. The evaluation indicators used in this experiment are defined as follows.

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

$$N_{tr} = \sum_{n=1}^{N}\left(M - \sum_{m=1}^{M}\left|\overline{y}^m - [y^m]\right|\right)/N, \tag{41}$$

where N denotes the total number of training trials, if $y^m \geq 0.5$, then $[y^m] = 1$, otherwise $[y^m] = 0$. Similarly, the number of correct prediction results for the testing set is defined as

$$N_{te} = \sum_{n=1}^{N}\left(\overline{M} - \sum_{m=1}^{\overline{M}}\left|\overline{y}^m - [y^m]\right|\right)/N, \tag{42}$$

where $\overline{M}$ denotes the number of samples in testing set.

The ratio of correct prediction results The ratio of correct prediction results for the training set is defined as

$$R_{tr} = 100N_{tr}/M. \tag{43}$$

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

$$R_{te} = 100N_{te}/\overline{M}. \tag{44}$$

Then, we use these four indicators and the average running time T_{avg} to compare QNNSIs with ANN. Experimental result contrasts are shown in Table 14.

It can be seen from Table 14 that the average running time of QNNSI85_1 is the shortest, and so, it is the most efficient. The R_{te} of QNNSI17_5 is the greatest, and so, its generalization ability is the strongest. For ANN85, although the R_{tr} is the greatest of the five models, its generalization ability is inferior to QNNSI17_5 and QNNSI5_17. In addition, for the four QNNSIs, almost all of the R_{te} are greater than the corresponding R_{tr}, which suggests that QNNSI has stronger generalization ability than ANN.

5.4 Breast cancer prediction

In this experiment, we give an example of predicting breast cancer with QNNSI and ANN. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at the following url: http://www.cs.wisc.edu/~street/images/. The dataset is linearly separable using all 30 input features, and 2 prediction fields respectively are benign and malignant. The number of instances in the dataset equals to 569, where 357 instances are benign and 212 instances are malignant. The best predictive accuracy obtained using one separating plane in the 3-D space of Worst Area, Worst Smoothness and Mean Texture. Separating plane described above can be obtained using Multi-surface Method-Tree (MSM-T), a classification method which uses linear programming to construct a decision tree. The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in Ref. [23]. The above-mentioned classifier has correctly diagnosed 176 consecutive new patients as of November 1995.

Table 14 The performance contrast of QNNSIs and ANN for caravan insurance policy prediction

Model	Index	Hidden nodes										
		10	11	12	13	14	15	16	17	18	19	20
QNNSI1_85	N_{tr}	4837.1	5003.2	4752.8	4756.1	4891.3	4941.8	4732.1	4772.8	4864.6	4774.7	4856.7
	R_{tr} (%)	83.083	85.937	81.635	81.692	84.015	84.881	81.280	81.978	83.556	82.011	83.419
	N_{te}	3326.1	3440.5	3266.2	3268.2	3363.4	3396.0	3253.3	3282.2	3344.4	3283.2	3336.8
	R_{te} (%)	**83.153**	**86.012**	**81.656**	**81.705**	**84.085**	**84.900**	**81.332**	**82.055**	**83.609**	**82.081**	**83.420**
	T_{avg} (s)	**23508**	**24329**	**27319**	**30642**	**33947**	**36282**	**38174**	**39897**	**42842**	**44537**	**47061**
QNNSI5_17	N_{tr}	5359.3	5472.7	5243.7	5467.5	5355.6	5281.0	4612.1	4630.4	5305.9	5471.2	4454.4
	R_{tr} (%)	92.053	94.000	90.067	93.911	91.989	90.708	79.218	79.533	91.135	93.975	76.510
	N_{te}	3682.4	3761.5	3602.3	3756.1	3682.6	3627.7	3163.6	3185.0	3648.4	3759.9	3062.6
	R_{te} (%)	**92.060**	**94.038**	**90.058**	**93.903**	**92.065**	**90.693**	**79.090**	**79.625**	**91.210**	**93.998**	**76.565**
	T_{avg} (s)	**3254.1**	**3506.2**	**3879.8**	**4132.0**	**4586.4**	**4838.0**	**5002.8**	**5291.1**	**5518.8**	**5804.1**	**5972.3**
QNNSI17_5	N_{tr}	5302.1	5473.8	5473.5	5439.1	5474.0	5471.2	5473.6	5473.9	5474.0	5312.4	5474.0
	R_{tr} (%)	91.070	94.019	94.014	93.423	94.023	93.975	94.016	94.021	94.023	91.247	94.023
	N_{te}	3646.2	3762.0	3762.0	3731.8	3762.0	3760.8	3762.0	3762.0	3762.0	3654.1	3761.9
	R_{te} (%)	**91.155**	**94.050**	**94.050**	**93.295**	**94.050**	**94.020**	**94.050**	**94.050**	**94.050**	**91.353**	**94.048**
	T_{avg} (s)	**1841.4**	**1995.2**	**2148.3**	**2343.3**	**2586.2**	**2833.7**	**2939.3**	**3188.7**	**3349.7**	**3600.3**	**3719.5**
QNNSI85_1	N_{tr}	2934.5	2864.3	2868.3	2923.4	2941.8	2873.4	2979.2	2918.5	2960.0	2875.8	2920.1
	R_{tr} (%)	50.040	49.198	49.267	50.213	50.529	49.354	51.171	50.129	50.842	49.395	50.156
	N_{te}	2025.6	1978.8	1975.1	2018.5	2019.8	1980.3	2056.9	1972.0	2035.4	1988.5	2004.9
	R_{te} (%)	**50.640**	**49.470**	**49.378**	**50.463**	**50.495**	**49.508**	**51.423**	**49.300**	**50.885**	**49.713**	**50.123**
	T_{avg} (s)	**1590.8**	**1747.6**	**1878.7**	**2078.1**	**2300.1**	**2318.5**	**2640.9**	**2693.1**	**3009.2**	**3096.7**	**3295.9**
ANN85	N_{tr}	5689.7	5715.3	5660.7	5666.8	5726.3	5735.6	5682.9	5711.8	5717.3	5687.9	5680.9
	R_{tr} (%)	97.728	98.167	97.229	97.334	98.356	98.516	97.611	98.107	98.202	97.697	97.576
	N_{te}	3612.5	3597.7	3639.3	3624.7	3586.2	3586.0	3616.6	3595.1	3591.3	3620.4	3627.6
	R_{te} (%)	**90.313**	**89.943**	**90.983**	**90.618**	**89.655**	**89.650**	**90.415**	**89.878**	**89.783**	**90.510**	**90.690**
	T_{avg} (s)	**1834.3**	**1994.2**	**2246.6**	**2332.1**	**2566.1**	**2802.1**	**2881.3**	**3130.3**	**3315.0**	**3575.6**	**3675.8**

Table 15 The input nodes and the sequence length setting of QNNSIs and ANN

QNNSIs		ANN	
Input nodes	Sequence length	Input nodes	Sequence length
1	30	30	1
2	15	30	1
3	10	30	1
5	6	30	1
6	5	30	1
10	3	30	1
15	2	30	1
30	1	30	1

In all samples, we use the first 400 instances (where 227 are benign) to train the network, and the remaining 169 instances (where 130 are benign) to test the generalization of the proposed model. For the input nodes and the sequence length, we employ the eight kinds of settings shown in Table 15. In this experiment, we set the number of hidden nodes to $5, 6, \ldots, 15$, respectively. The maximum number of iterative steps is set to 100. The other parameters of QNNSIs and ANN are set in the same way as the previous experiment. In this experiment, we do not set the value of the target error. The algorithm is not stopped until it reaches the predefined maximum number of iterative steps.

Our experiment scheme is below. QNNSIs and ANN train 10 times for each setting of hidden nodes by the training set data, and are immediately tested by the testing set data after each training. Then, we use the same evaluation indicators as the previous experiment to compare QNNSIs with ANN. Experimental contrast results are shown in Table 16.

It can be seen from Table 16 that, as far as the approximation and generalization ability are concerned, QNNSI1_30 and QNNSI30_1 are obviously inferior to ANN30; QNNSI2_15 and QNNSI15_2 are roughly equal to ANN30; QNNSI3_10 and QNNSI10_3 are slightly superior to ANN30; QNNSI5_6 and QNNSI6_5 are obviously superior to ANN30. As far as the average running time are concerned, QNNSI1_30, QNNSI2_15, and QNNSI3_10 are obviously longer than ANN30; QNNSI5_6 and QNNSI6_5 are slightly longer than ANN30; QNNSI10_3, QNNSI15_2, and QNNSI30_1 are roughly equal to ANN30. Synthesizing the above-mentioned two aspects, QNNSI shows better performance than ANN when the number of input nodes is close to the sequence length.

Next, we theoretically explain the above experimental results. Assume that n denotes the number of input nodes, q denotes the sequence length, p denotes the number of hidden nodes, and m denotes the number of output nodes, and the product of nq is approximately a constant.

It is clear that the number of adjustable parameters in QNNSI and ANN is the same, i.e., equals $npq + pm$. The weights adjustment formula in the output layer of QNNSI and ANN is also the same. But, their parameters adjustment of hidden layer is completely different. The adjustment of hidden parameters in QNNSI is much more complex than that in ANN. In ANN, each hidden parameter adjustment only involves two derivative calculations. In QNNSI, each hidden layer parameter adjustment involves at least two and at most $q + 1$ derivative calculations.

In QNNSI, when $q = 1$, although the number of input nodes is the greatest possible, the calculation of the hidden layer output and hidden parameter adjustment are also the most simple, which directly lead to the reduction of the approximation ability. When $n = 1$, the calculation of the hidden layer output is the most complex, which make the QNNSI have the strongest nonlinear mapping ability. However, at this time, the calculation of hidden parameter adjustment is also very complex. A large number of derivative calculations can lead to the adjustment of parameters which tend to zero or infinity. This can hinder the convergence of the training process and lead to the reduction of the approximation ability. Hence, when $q = 1$ or $n = 1$, the approximation ability of QNNSI is inferior to that of ANN. When $n > 1$ or $q > 1$, the approximation ability of QNNSI tends to improve, and under a certain condition, the approximation ability of QNNSI will certainly be superior to that of ANN. The above analysis is consistent with the experimental results.

In addition, what is the accurate relationship between n and q to make QNNSI approximation ability the strongest? This problem needs further study, and usually depends on the specific issues. Our conclusions based on experiments is as follows: when $q/2 \le n \le 2q$, QNNSIn_q is superior to the ANN with nq input nodes.

It is worth pointing out that QNNSI 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 QNNSI, 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 used in QNNSI are concerned, information processing can be performed simultaneously, which greatly improves the computational efficiency. Because the above four experiments are performed

Table 16 The performance contrast of QNNSIs and ANN for breast cancer prediction

Model	Index	Hidden nodes										
		5	6	7	8	9	10	11	12	13	14	15
QNNSI1_30	N_{tr}	180.8	217.0	188.1	210.5	198.2	237.2	202.7	219.5	198.2	203.0	204.9
	R_{tr} (%)	45.20	54.25	47.03	52.63	49.55	59.30	50.68	54.88	49.55	50.75	51.23
	N_{te}	66.00	86.60	53.00	92.70	66.20	95.30	64.60	92.40	65.60	58.30	68.40
	R_{te} (%)	**39.05**	**51.24**	**31.36**	**54.85**	**39.17**	**56.39**	**38.22**	**54.67**	**38.82**	**34.50**	**40.47**
	T_{avg} (s)	**204.8**	**253.0**	**312.9**	**327.8**	**333.8**	**360.6**	**422.4**	**455.7**	**502.5**	**536.4**	**585.3**
QNNSI2_15	N_{tr}	325.2	342.9	359.4	347.9	338.8	320.3	314.6	309.2	248.3	239.1	178.4
	R_{tr} (%)	81.30	85.73	89.85	86.98	84.70	80.08	78.65	77.30	62.08	59.78	44.60
	N_{te}	132.4	130.6	149.9	138.3	142.5	132.0	135.2	123.8	79.70	99.90	48.10
	R_{te} (%)	**78.34**	**77.28**	**88.70**	**81.83**	**84.32**	**78.11**	**80.00**	**73.25**	**47.16**	**59.11**	**28.46**
	T_{avg} (s)	**89.33**	**115.7**	*122*.**5**	**145.1**	**181.6**	**217.2**	**240.0**	**317.1**	**359.6**	**275.8**	**281.8**
QNNSI3_10	N_{tr}	378.5	366.9	378.6	392.8	392.2	378.0	356.7	378.2	369.2	360.8	362.5
	R_{tr} (%)	94.63	91.73	94.65	98.20	98.05	94.50	89.18	94.55	92.30	90.20	90.63
	N_{te}	148.0	136.2	148.1	161.2	156.6	158.5	143.2	145.5	151.9	151.5	138.1
	R_{te} (%)	**87.57**	**80.59**	**87.63**	**95.38**	**92.66**	**93.79**	**84.73**	**86.09**	**89.88**	**89.65**	**81.72**
	T_{avg} (s)	**77.45**	**82.46**	**108.4**	**119.3**	**141.3**	**174.7**	**172.8**	**182.2**	**222.1**	**218.1**	**222.1**
QNNSI5_6	N_{tr}	396.1	397.5	396.5	397.4	394.9	397.3	396.9	397.8	397.7	399.0	398.3
	R_{tr} (%)	99.03	99.38	99.13	99.35	98.73	99.33	99.23	99.45	99.43	99.75	99.58
	N_{te}	162.1	164.6	162.4	162.0	160.0	162.8	159.9	161.6	163.1	162.1	161.3
	R_{te} (%)	**95.92**	**97.40**	**96.09**	**95.86**	**94.67**	**96.33**	**94.62**	**95.62**	**96.51**	**95.92**	**95.44**
	T_{avg} (s)	**51.35**	**63.32**	**69.94**	**86.37**	**110.6**	**118.8**	**137.1**	**134.9**	**170.8**	**172.9**	**214.6**
QNNSI6_5	N_{tr}	396.5	397.8	397.7	395.5	395.7	398.0	397.4	397.4	398.0	398.0	398.0
	R_{tr} (%)	99.13	99.45	99.43	98.88	98.93	99.50	99.35	99.35	99.50	99.50	99.50
	N_{te}	163.4	164.0	162.4	162.8	164.4	162.0	164.9	163.9	163.4	163.8	162.4
	R_{te} (%)	**96.69**	**97.04**	**96.09**	**96.33**	**97.28**	**95.86**	**97.57**	**96.98**	**96.69**	**96.92**	**96.09**
	T_{avg} (s)	**49.18**	**67.04**	**68.45**	**92.39**	**94.67**	**115.0**	**147.0**	**146.2**	**161.7**	**154.5**	**209.9**
QNNSI10_3	N_{tr}	373.5	395.5	366.7	382.0	393.2	392.2	356.8	387.6	392.8	393.6	392.6
	R_{tr} (%)	93.38	98.88	91.68	95.50	98.30	98.05	89.20	96.90	98.20	98.40	98.15
	N_{te}	156.5	160.2	146.2	153.7	156.8	155.5	148.1	159.2	159.8	161.2	159.3
	R_{te} (%)	**92.60**	**94.79**	**86.51**	**90.95**	**92.78**	**92.01**	**87.63**	**94.20**	**94.56**	**95.38**	**94.26**
	T_{avg} (s)	**42.44**	**59.02**	**62.93**	**83.88**	**82.65**	**104.3**	**110.5**	**134.6**	**136.1**	**166.8**	**196.0**
QNNSI15_2	N_{tr}	278.8	247.4	348.6	276.5	360.9	368.1	360.2	356.6	341.8	393.9	375.5
	R_{tr} (%)	69.70	61.85	87.15	69.13	90.23	92.03	90.05	89.15	85.45	98.48	93.88
	N_{te}	126.0	121.0	147.6	111.3	151.6	158.5	156.4	153.7	150.0	162.9	162.3
	R_{te} (%)	**74.56**	**71.59**	**87.34**	**65.86**	**89.70**	**93.79**	**92.54**	**90.95**	**88.76**	**96.39**	**96.04**
	T_{avg} (s)	**42.22**	**44.65**	**60.23**	**62.27**	**86.29**	**104.6**	**104.5**	**138.9**	**134.9**	**151.3**	**148.5**
QNNSI30_1	N_{tr}	229.3	203.9	226.4	222.9	209.1	209.0	231.6	216.0	214.5	235.4	200.7
	R_{tr} (%)	57.33	50.98	56.60	55.73	52.28	52.25	57.90	54.00	53.63	58.85	50.18
	N_{te}	105.9	90.10	104.9	84.10	93.10	88.10	100.0	84.00	100.0	98.10	84.20
	R_{te} (%)	**62.66**	**53.31**	**62.07**	**49.76**	**55.09**	**52.13**	**59.17**	**49.70**	**59.17**	**58.05**	**49.82**
	T_{avg} (s)	**42.39**	**45.21**	**63.31**	**63.61**	**86.65**	**86.29**	**115.4**	**112.6**	**140.9**	**150.4**	**176.5**
ANN30	N_{tr}	364.6	274.1	342.0	347.4	324.8	336.6	347.4	365.9	359.2	376.4	359.2
	R_{tr} (%)	91.15	68.53	85.50	86.85	81.20	84.15	86.85	91.48	89.80	94.10	89.80
	N_{te}	153.7	105.7	142.9	151.6	139.7	133.7	151.9	147.8	145.6	149.1	145.8
	R_{te} (%)	**90.95**	**62.54**	**84.56**	**89.70**	**82.66**	**79.11**	**89.88**	**87.46**	**86.15**	**88.22**	**86.27**
	T_{avg} (s)	**50.52**	**53.80**	**71.63**	**72.42**	**97.72**	**98.18**	**125.4**	**127.7**	**145.9**	**136.1**	**154.6**

in classical computer, the quantum parallelism has not been explored. However, the efficient computational ability of QNNSI is bound to stand out in future quantum computer.

6 Conclusions

This paper proposes a quantum-inspired neural network model with sequence input based on the principle of quantum computing. The architecture of the proposed model includes three layers, where the hidden layer consists of quantum neurons and the output layer consists of classical neurons. An obvious difference from classical ANN is that each dimension of a single input sample consists of a discrete sequence rather that a single value. The activation function of hidden layer is redesigned according to the principle of quantum computing. The *Levenberg–Marquardt* algorithm is employed for learning. With the application of the information processing mechanism of quantum controlled-rotation gates, the proposed model can effectively obtain the sample characteristics by way of *breadth* and *depth*. The experimental results reveal that a greater difference between input nodes and sequence length leads to a lower performance of the proposed model than that of the classical ANN, on the contrary, it obviously enhances the approximation and generalization ability of the proposed model when input nodes are closer to the sequence length. The following issues to the proposed model, such as continuity, computational complexity, and improvement of the learning algorithm, are subject of further research.

Acknowledgements We thank the three anonymous reviewers sincerely for their many constructive comments and suggestions, which have tremendously improved the presentation and quality of this paper. This work was supported by the National Natural Science Foundation of China (Grant No. 61170132).

Appendix: The gradient calculation in Levenberg–Marquardt algorithm

According to the gradient descent algorithm in Ref. [22], the gradient of the rotation angles of the quantum rotation gates in hidden layer and the connection weights in output layer can be calculated as follows

$$\begin{cases} \dfrac{\partial e_k^l}{\partial h_j^l(t_q)} = -y_k^l(1-y_k^l)w_{jk}, \\ \dfrac{\partial h_j^l(t_r)}{\partial h_j^l(t_{r-1})} = (V_{jr}^l + \overline{V}_{jr}^l)/\left(2h_j^l(t_r)\sqrt{1-(h_j^l(t_{r-1}))^2}\right), \\ \dfrac{\partial h_j^l(t_r)}{\partial \theta_{ij}(t_r)} = \begin{cases} \dfrac{(\overline{h}_{jr}^l)^2(U_{jr}^l+\overline{U}_{jr}^l)\cot(\theta_i^l(t_r)+\theta_{ij}(t_r))}{h_j^l(t_r)}, \\ \quad j=1,3,5,\ldots, \\ -\dfrac{(\overline{h}_{jr}^l)^2(U_{jr}^l+\overline{U}_{jr}^l)\tan(\theta_i^l(t_r)+\theta_{ij}(t_r))}{h_j^l(t_r)}, \\ \quad j=2,4,6,\ldots, \end{cases} \\ \dfrac{\partial h_j^l(t_r)}{\partial \overline{\varphi}_j(t_r)} = \dfrac{(\overline{h}_{jr}^l)^2(W_{jr}^l+\overline{W}_{jr}^l)}{h_j^l(t_r)}, \end{cases} \tag{45}$$

where

$$\begin{cases} \overline{h}_{jr}^l = \begin{cases} \prod_{i=1}^n \sin(\theta_i^l(t_r)+\theta_{ij}(t_r)) & j=1,3,5,\ldots, \\ \prod_{i=1}^n \cos(\theta_i^l(t_r)+\theta_{ij}(t_r)) & j=2,4,6,\ldots, \end{cases} \\ U_{jr}^l = h_j^l(t_{r-1})\sqrt{1-\left(h_j^l(t_{r-1})\right)^2}\sin\left(2\overline{\varphi}_j(t_r)\right), \\ \overline{U}_{jr}^l = \left(1-2\left(h_j^l(t_{r-1})\right)^2\right)\sin^2\left(\overline{\varphi}_j(t_r)\right), \\ V_{jr}^l = \left(1-2\left(h_j^l(t_{r-1})\right)^2\right)\left(\overline{h}_{jr}^l\right)^2\sin\left(2\overline{\varphi}_j(t_r)\right), \\ \overline{V}_{jr}^l = 2h_j^l(t_{r-1})\left(1-2\left(\overline{h}_{jr}^l\right)^2\sin^2\left(\overline{\varphi}_j(t_r)\right)\right) \\ \qquad \times\sqrt{1-\left(h_j^l(t_{r-1})\right)^2}, \\ W_{jr}^l = h_j^l(t_{r-1})\sqrt{1-\left(h_j^l(t_{r-1})\right)^2}\cos\left(2\overline{\varphi}_j(t_r)\right), \\ \overline{W}_{jr}^l = 0.5\left(1-2\left(h_j^l(t_{r-1})\right)^2\right)\sin\left(2\overline{\varphi}_j(t_r)\right). \end{cases} \tag{46}$$

Based on the above Eq. (45), we obtain

$$\begin{aligned} \frac{\partial e_k^l}{\partial \theta_{ij}(t_r)} &= \frac{\partial e_k^l}{\partial h_j^l(t_q)}\cdots\frac{\partial h_j^l(t_{r+1})}{\partial h_j^l(t_r)}\frac{\partial h_j^l(t_r)}{\partial \theta_{ij}(t_r)} \\ &= \begin{cases} (-y_k^l(1-y_k^l)w_{jk}\prod_{s=r+1}^q(V_{js}^l+\overline{V}_{js}^l)(\overline{h}_{jr}^l)^2 \\ \quad\times(U_{jr}^l+\overline{U}_{jr}^l)\cot(\theta_i^l(t_r)+\theta_{ij}(t_r))) \\ \quad\times\left(h_j^l(t_r)\prod_{s=r+1}^q 2h_j^l(t_s)\sqrt{1-(h_j^l(t_{s-1}))^2}\right)^{-1} \\ \quad j=1,3,5,\ldots, \\ (y_k^l(1-y_k^l)w_{jk}\prod_{s=r+1}^q(V_{js}^l+\overline{V}_{js}^l)(\overline{h}_{jr}^l)^2 \\ \quad\times(U_{jr}^l+\overline{U}_{jr}^l)\tan(\theta_i^l(t_r)+\theta_{ij}(t_r))) \\ \quad\times\left(h_j^l(t_r)\prod_{s=r+1}^q 2h_j^l(t_s)\sqrt{1-(h_j^l(t_{s-1}))^2}\right)^{-1} \\ \quad j=2,4,6,\ldots, \end{cases} \end{aligned} \tag{47}$$

$$\frac{\partial e_k^l}{\partial \overline{\varphi}_j(t_r)} = \frac{\partial e_k^l}{\partial h_j^l(t_q)} \cdots \frac{\partial h_j^l(t_{r+1})}{\partial h_j^l(t_r)} \frac{\partial h_j^l(t_r)}{\partial \overline{\varphi}_j(t_r)}$$
$$= \left(-y_k^l(1-y_k^l) w_{jk} \times \prod_{s=r+1}^{q} (V_{js}^l + \overline{V}_{js}^l)(\overline{h}_{jr}^l)^2 (W_{jr}^l + \overline{W}_{jr}^l) \right) \times \left(h_j^l(t_r) \prod_{s=r+1}^{q} 2h_j^l(t_s)\sqrt{1-(h_j^l(t_{s-1}))^2} \right)^{-1}, \tag{48}$$

where $j = 1, 2, \ldots, p$, $k = 1, 2, \ldots, m$, $r = 1, 2, \ldots, q$, $l = 1, 2, \ldots, L$.

The gradient of the connection weights in the outer layer can be calculated as follows

$$\frac{\partial e_k^l}{\partial w_{jk}} = -y_k^l(1-y_k^l) h_j^l(t_q). \tag{49}$$

References

1. Nekoukar V, Beheshti MTH (2010) A local linear radial basis function neural network for financial time-series forecasting. Appl Intell 33:352–356
2. Lee H, Kim E, Pedrycz W (2012) A new selective neural network ensemble with negative correlation. Appl Intell 37:488–498
3. Park B-J, Pedrycz W, Oh S-K (2010) Polynomial-based radial basis function neural networks (P-RBF NNs) and their application to pattern classification. Appl Intell 32:27–46
4. Hassan YF (2011) Rough sets for adapting wavelet neural networks as a new classifier system. Appl Intell 35:260–268
5. Kim K-j, Ahn H (2012) Simultaneous optimization of artificial neural networks for financial forecasting. Appl Intell 36:887–898
6. Tsoi AC, Back AD (1994) Locally recurrent globally feedforward networks: a critical review of architectures. IEEE Trans Neural Netw 5(2):229–239
7. Kleinfeld D (1986) Sequential state generation by model neural network. In: Proceedings of the national academy of sciences of the United States of America, pp 9469–9473
8. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang KJ (1989) Phoneme recognition using time-delay neural networks. IEEE Trans Acoust Speech Signal Process 37(3):328–339
9. Lippmann RP (1989) Review of neural network for speech recognition. Neural Comput 1(1):1–38
10. Moustra M, Avraamides M, Christodoulou C (2011) Artificial neural network for earthquake prediction using time series magnitude data or seismic electric signals. Expert Syst Appl 38:15032–15039
11. Kak S (1995) On quantum neural computing. Inf Sci 83(3–4):143–160
12. Purushothaman G, Karayiannis NB (1997) Quantum neural network (QNN's): inherently fuzzy feedforward neural network. IEEE Trans Neural Netw 8(3):679–693
13. Zak M, Williams CP (1998) Quantum neural nets. Int J Theor Phys 37(2):651–684
14. Maeda M, Suenaga M, Miyajima H (2007) Qubit neuron according to quantum circuit for XOR problem. Appl Math Comput 185(2):1015–1025
15. Gupta S, Zia RKP (2001) Quantum neural network. J Comput Syst Sci 63(3):355–383
16. Shafee F (2007) Neural network with quantum gated nodes. Eng Appl Artif Intell 20(4):429–437
17. Li PC, Li SY (2008) Learning algorithm and application of quantum BP neural network based on universal quantum gates. J Syst Eng Electron 19(1):167–174
18. Li PC, Song KP, Yang EL (2010) Model and algorithm of neural network with quantum gated nodes. Neural Netw World 11(2):189–206
19. da Silva AJ, de Oliveira WR, Ludermir TB (2012) Classical and superposed learning for quantum weightless neural network. Neurocomputing 75(1):52–60
20. Gonzalez-Carrasco I, Garcia-Crespo A, Ruiz-Mezcua B, Lopez-Cuadrado JL (2011) Dealing with limited data in ballistic impact scenarios: an empirical comparison of different neural network approaches. Appl Intell 35:89–109
21. Gonzalez-Carrasco I, Garcia-Crespo A, Ruiz-Mezcua B, Lopez-Cuadrado JL (2012) An optimization methodology for machine learning strategies and regression problems in ballistic impact scenarios. Appl Intell 36:424–441
22. Hagan MT, Demuth HB, Beale MH (1996) Neural network design. PWS Publishing Company, USA
23. Bennetta KP, Mangasariana OL (1992) Robust linear programming discrimination of two linearly inseparable sets. Optim Methods Softw 1(1):23–34

Panchi Li received the B.S. and M.S. degrees from Northeast Petroleum University, China, in 1998 and 2004, respectively, and the Ph.D. degree from Harbin Institute of Technology, China, in 2009. Currently, he is a professor in the school of computer and information technology, Northeast Petroleum University, China. His current research interests include quantum neural networks and quantum optimization algorithms.

Hong Xiao received the B.S. and M.S. degrees from Northeast Petroleum University, China, in 2001 and 2004, respectively, and then she became a teacher there. Currently, she is a Ph.D. candidate in Northeast Petroleum University, China. Her current research interests include neural networks and evolutionary algorithms.