EasyChair Preprint

№ 1108

Predicting PPI Based on Quantum-inspired Neural Networks

Li-Ping Yang, Cheng Zhang and Li Qin

June 8, 2019

Predicting PPI Based on Quantum-inspired Neural Networks

Li-Ping Yang
College of Informatics
Huazhong Agricultural University
Wuhan, China
teresa@mail.hzau.edu.cn

Cheng Zhang*
College of Informatics
Huazhong Agricultural University
Wuhan, China
sameboy@mail.hzau.edu.cn

Li Qin
College of Informatics
Huazhong Agricultural University
Wuhan, China
qinli@mail.hzau.edu.cn

Abstract*—Studying how and when PPI (Protein-Protein Interaction) happens in cells is very important for catching molecules mechanism during life. Unfortunately, present classical computers technology and process speed for massive PPI data is far from meeting demand. It is a general trend of researching that utilizing quantum computation methods, bioinformatics knowledge and machine learning ability to effectively investigate existing massive data for discovering and verifying new PPI. In this paper, we utilize the quantum ant colony optimization algorithm (QAC) to optimize the quantum-inspired neural network QNN's parameters and propose a novel quantum neurocomputing scheme QAC-QNN which can prevent the solution sinking into local optimization. The simulation experiments prove that the novel scheme is feasible. For predicting PPI of proteins couple, in contrast with SVM and ANN, QAC-QNN achieves better prediction effects.

Keywords*—quantum computation, neural network, ant colony optimization, PPI, bioinformatics

I. INTRODUCTION

Research has shown that almost all proteins cannot perform functions alone, but they function through their interaction with other proteins. Cell's reaction to change of external environment and internal environment is all through the link: protein-proteins interaction (in this paper, we use the abbreviation *PPI* for denoting it). Therefore, studying how and when PPI happens in cells is very important for catching molecules mechanism during life and grasping molecules basis of diseases.

Widespread use of calculation methods in biology develops a vaster space of PPI prediction. Currently, a lot of PPI prediction algorithms from the view of different biological knowledge surge. Many of them construct prediction models based on genome information, protein family evolution information, protein sequence information or protein structure information[1,2]. Some of them mix multiple information mentioned above together[3,4,5] to predict PPI. Methods based on genome information are shown as follows. In 1999, Pellegrini[6] et al. proposed a phylogenetic profile method. In the same year, Enright et al. proposed gene fusion event. Methods based on protein's first level acid residues sequence or structure information are shown as follows. In 2001, Sprinzak[7] et al. proposed a correlated sequence-signatures method. In 2005, Martin[8] et al. designed a kind of characteristic description symbol about protein amino acid sequence to predict PPI. In 2008, Guo[9] et al. proposed an encoding method using autocovariance of amino acid sequence to predict yeast PPI.

However, the amount of PPI data in the current existing biology database is tremendous. For example, there are almost 70 thousand PPI records about 24430 proteins or so. Unfortunately, present classical computers technology and process speed for such massive data is far from meeting demand. Therefore, it is a general trend of researching that utilizing quantum computation methods, bioinformatics knowledge and machine learning ability to effectively investigate existing massive data for discovering and verifying new PPI.

This paper draws lessons from the idea of Panchi Li's Quantum-inspired Neural Network[10] which is called simply as "QNN" below. We combine biological characteristic of the application problem which will be explained in Section Ⅳ with QNN, design each level and its parameters of the neural network. Moreover, this paper

adopts idea of the quantum ant colony optimization algorithm which is proposed by Panchi Li[11]. This algorithm will be introduced in Part B of Section II and called simply as "QAC" below. We then use QAC to train the parameters of our specified QNN in order to speed convergence of our method.

II. RELATED WORK

A. Quantum-inspired Neural Networks

In literature [10], the Quantum-inspired neuron has been proposed. Quantum bits are inputted into the Quantum-inspired neuron, the rotation angles are considered as the parameters to be adjusted, and real numbers are outputted from the neuron.

The mapping of this neuron is as explained as follows. Initially, the input quantum bits $|x_i\rangle (i=1,2,\cdots,n)$ revolve angle δ_i round their X, Y and Z axes respectively. Then, the quantum bits are measured by the projective measurements right now, and the obtained coordinate values are added up afterwards. Finally, the output of this neuron is achieved through activation functions.

Let $|x_i\rangle = \left[\cos(\theta_i/2) \quad e^{i\phi_i}\sin(\theta_i/2)\right]^T$ and we assume that $|\tilde{x}_i\rangle, |\tilde{y}_i\rangle$ and $|\tilde{z}_i\rangle$ are obtained after $|x_i\rangle$ revolves round X, Y and Z axes respectively. So, the rotation in which $|x_i\rangle$ revolves angle δ_i round their X, Y and Z axes can be expressed as follows:

$$
\begin{aligned}
|\tilde{x}_i\rangle &= R_x(\delta_i)|x_i\rangle \\
&= \begin{bmatrix} \cos\frac{\delta_i}{2} & -i\sin\frac{\delta_i}{2} \\ -i\sin\frac{\delta_i}{2} & \cos\frac{\delta_i}{2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta_i}{2} \\ e^{i\phi_i}\sin\frac{\theta_i}{2} \end{bmatrix} \\
&= \begin{bmatrix} \cos\frac{\delta_i}{2}\cos\frac{\theta_i}{2} - ie^{i\phi_i}\sin\frac{\delta_i}{2}\sin\frac{\theta_i}{2} \\ -i\sin\frac{\delta_i}{2}\cos\frac{\theta_i}{2} + e^{i\phi_i}\cos\frac{\delta_i}{2}\sin\frac{\theta_i}{2} \end{bmatrix}
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
|\tilde{y}_i\rangle &= R_y(\delta_i)|x_i\rangle \\
&= \begin{bmatrix} \cos\frac{\delta_i}{2} & -\sin\frac{\delta_i}{2} \\ \sin\frac{\delta_i}{2} & \cos\frac{\delta_i}{2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta_i}{2} \\ e^{i\phi_i}\sin\frac{\theta_i}{2} \end{bmatrix} \\
&= \begin{bmatrix} \cos\frac{\delta_i}{2}\cos\frac{\theta_i}{2} - e^{i\phi_i}\sin\frac{\delta_i}{2}\sin\frac{\theta_i}{2} \\ \sin\frac{\delta_i}{2}\cos\frac{\theta_i}{2} + e^{i\phi_i}\cos\frac{\delta_i}{2}\sin\frac{\theta_i}{2} \end{bmatrix}
\end{aligned} \tag{2}
$$

$$
|\tilde{z}_i\rangle = R_z(\delta_i)|x_i\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta_i} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta_i}{2} \\ e^{i\phi_i}\sin\frac{\theta_i}{2} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta_i}{2} \\ e^{i(\delta_i+\phi_i)}\sin\frac{\theta_i}{2} \end{bmatrix} \tag{3}
$$

The coordinates of $|\tilde{x}_i\rangle, |\tilde{y}_i\rangle$ and $|\tilde{z}_i\rangle$ on the Bloch sphere are shown in Eq. (4)--(6) , where $\langle\bullet|$ is the conjugate transpose of $|\bullet\rangle$.

It is well found in Eq. (4)--(6) that the coordinates of $|x_i\rangle$ don't change after it revolves round X, Y and Z axes. Consequently, when these coordinates are added up, the unchanged coordinate values can be ignored.

$$
\begin{aligned}
x\_\tilde{x}_i &= \langle\tilde{x}_i|\sigma_x|\tilde{x}_i\rangle = \sin\theta_i\cos\phi_i \\
y\_\tilde{x}_i &= \langle\tilde{x}_i|\sigma_y|\tilde{x}_i\rangle = \sin\theta_i\sin\phi_i\cos\delta_i - \cos\theta_i\sin\delta_i \\
z\_\tilde{x}_i &= \langle\tilde{x}_i|\sigma_z|\tilde{x}_i\rangle = \sin\theta_i\sin\phi_i\sin\delta_i + \cos\theta_i\cos\delta_i
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
x\_\tilde{y}_i &= \langle\tilde{y}_i|\sigma_x|\tilde{y}_i\rangle = \sin\theta_i\cos\phi_i\cos\delta_i + \cos\theta_i\sin\delta_i \\
y\_\tilde{y}_i &= \langle\tilde{y}_i|\sigma_y|\tilde{y}_i\rangle = \sin\theta_i\sin\phi_i \\
z\_\tilde{y}_i &= \langle\tilde{y}_i|\sigma_z|\tilde{y}_i\rangle = \cos\theta_i\cos\delta_i - \sin\theta_i\cos\phi_i\sin\delta_i
\end{aligned} \tag{5}
$$

$$
\begin{aligned}
x\_\tilde{z}_i &= \langle\tilde{z}_i|\sigma_x|\tilde{z}_i\rangle = \cos(\phi_i+\delta_i)\sin\theta_i \\
y\_\tilde{z}_i &= \langle\tilde{z}_i|\sigma_y|\tilde{z}_i\rangle = \sin(\phi_i+\delta_i)\sin\theta_i \\
z\_\tilde{z}_i &= \langle\tilde{z}_i|\sigma_z|\tilde{z}_i\rangle = \cos\theta_i
\end{aligned} \tag{6}
$$

Let $V_i = 0.5(y\_\tilde{x}_i + z\_\tilde{x}_i + x\_\tilde{y}_i + z\_\tilde{y}_i + x\_\tilde{z}_i + y\_\tilde{z}_i)$, it can be achieved through Eq. (4)--(6) that:

$$
V_i = (\sin\theta_i\cos\phi_i + \sin\theta_i\sin\phi_i + \cos\theta_i)\cos\delta_i \tag{7}
$$

The output of quantum neuron in hidden layer is shown as Eq. (8), where the activation function is Sigmoid function.

$$y = \frac{1}{1+e^{-\sum_{i=1}^{n} V_i}} \tag{8}$$

The quantum neural networks model proposed by Panchi Li is shown as Fig. 1. QNN is composed of an input layer, a hidden layer and an output layer. The input layer is composed of qubits; the hidden layer is composed of quantum-inspired neurons; the output layer is composed of artificial neurons. The number of neurons is as follows. There are *n* neurons in input layer, *p* neurons in hidden layer, and *m* neurons in output layer. Activation function in output layer also is Sigmoid function.

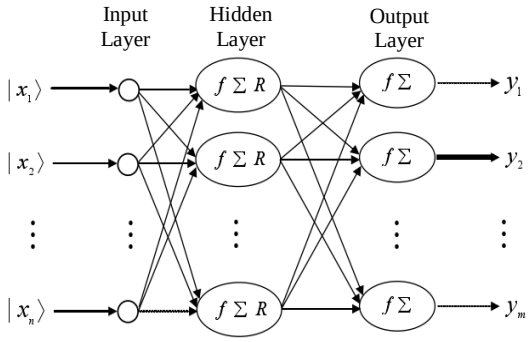

Fig. 1. Quantum neural networks model (QNN)[10]

The output of hidden layer and that of output layer can be calculated respectively through Eq. (9) and Eq. (10) .

$$h_j = f\left(\sum_{i=1}^{n} V_{ij}\right) = \frac{1}{1+e^{-\sum_{i=1}^{n}(\sin\theta_i \cos\phi_j + \sin\theta_i \sin\phi_j + \cos\theta_i)\cos\delta_{ij}}} \tag{9}$$

$$y_k = f\left(\sum_{j=1}^{p} \omega_{jk} h_j\right) = \frac{1}{1+e^{-\sum_{j=1}^{p} \omega_{jk} h_j}} \tag{10}$$

where j=1,2,⋯,p， k=1,2,⋯,m.

B. Quantum Ant Colony Optimization Algorithm

1) Principle of quantum ant colony optimization algorithm

The principle of quantum ant colony optimization algorithm is shown as follows. Let there are *m* ants in an ant colony, and they lie randomly in the n-dimensional unit space $\tilde{\Omega} = [-1,1]^n$. Each ant carries one group of *n* qubits whose probability amplitudes represent the current place of this ant in $\tilde{\Omega}$. In QAC algorithm, two amplitudes according to the two calculation ground states of the same qubit are considered as this ant's place information. Hence in $\tilde{\Omega}$, each ant can occupy two positions. A group of qubits which are carried by the ant *v* is denoted as follows.

$$X_v = \begin{vmatrix} \cos\varphi_{v1} & \cos\varphi_{v2} & \dots & \cos\varphi_{vn} \\ \sin\varphi_{v1} & \sin\varphi_{v2} & \dots & \sin\varphi_{vn} \end{vmatrix} \tag{11}$$

Therefore, the two positions occupied by the ant *v* are $(\cos\varphi_{v1}, \cdots, \cos\varphi_{vn})$ and $(\sin\varphi_{v1}, \cdots, \sin\varphi_{vn})$ respectively. Obviously, QAC can double the scales of space in which the solution will be obtained on condition that the total number of ants in the colony keeps constant. It can also double the convergence rate of optimization algorithm.

2) Framework of QAC

If denote the pheromone strength of the ant *v* in the position x_r by $\tau(x_r)$, and denote the visibility in the same position by $\eta(x_r)$, the main idea of QAC is shown as follows.

a) Select the ant's objective position to move

When the ant *v* moves from the source position x_r to the objective position x_s, it must obey the following rule:

$$x_s = \begin{cases} \arg\max_{x_s \in X} \{[\tau(x_s)]^\alpha [\eta(x_s)]^\beta\} & q \le q_0 \\ \tilde{x}_s & q > q_0 \end{cases} \tag{12}$$

$$p(x_s) = \frac{[\tau(x_s)]^\alpha [\eta(x_s)]^\beta}{\sum_{x_u \in X} [\tau(x_u)]^\alpha [\eta(x_u)]^\beta} \tag{13}$$

where $\tilde{x}_s$ represents the selected objective position according to Eq. (13). *X* is the set of all position points which all the ants occupy in $\tilde{\Omega}$. The *q* is a random figure which distributes evenly in the interval [0,1] and q_0 is an arbitrary constant which satisfies $0 \le q_0 \le 1$.

b) The ant moves to objective position

In the *step a)*, the ant *v* has started to move to the objective position x_s. The ant relies on quantum rotation gates to change the phase of its qubit to finish moving.

Let current position of the ant *v* is x_r and its objective position is x_s. Then their corresponding qubits are shown as follows.

$$X_r = \begin{vmatrix} \cos\varphi_{r1} & \cos\varphi_{r2} & \dots & \cos\varphi_{rn} \\ \sin\varphi_{r1} & \sin\varphi_{r2} & \dots & \sin\varphi_{rn} \end{vmatrix} \tag{14}$$

$$X_s = \begin{vmatrix} \cos\varphi_{s1} & \cos\varphi_{s2} & \dots & \cos\varphi_{sn} \\ \sin\varphi_{s1} & \sin\varphi_{s2} & \dots & \sin\varphi_{sn} \end{vmatrix} \tag{15}$$

In order to apply a quantum rotation gate to X_r to make it approximate to X_s, it is critical that choosing appropriate direction and size of the rotation gate's angle $\Delta\theta$. The direction of rotation angle can be chosen according to the following rules. Let probability amplitudes of the objective qubit are α_0 and β_0; amplitudes of current qubit are α_1 and β_1. Denote

$$A = \begin{vmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{vmatrix} \tag{16}$$

When A equals 0, positive or negative direction is all right; when A doesn't equal 0, select $-\mathrm{sgn}(A)$ as the direction. At present, there isn't theory basis about what size of the rotation angle should be selected. Panchi Li[11] has proposed a strategy about the rotation angle is as follow.

$$\Delta\theta = -\mathrm{sgn}(A)\theta_0 e^{-t} \tag{17}$$

where $\theta_0 \in (0.001\pi, 0.05\pi)$ is the initial value of iteration, and t is the number of optimization steps.

c) Implementing mutation

In order to avoid the solution of QAC sinking into local optimization, quantum NOT gate is selected to implement mutation for ants' current positions. At first, choose several ants from colony randomly. Then choose several qubits from these ants' qubits randomly. In order to exchange the two amplitudes of each selected qubit, it is operated by a quantum NOT gate. Let the i-th qubit which the ant v carries is $\left[\alpha_{vi} \quad \beta_{vi}\right]^T$, the mutation operation is as follows.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_{vi} \\ \beta_{vi} \end{bmatrix} = \begin{bmatrix} \beta_{vi} \\ \alpha_{vi} \end{bmatrix} \tag{18}$$

The mutation operation mentioned above is advantageous to diversity of ant colony and preventing premature convergence of solution.

d) Updating ants' pheromone strength values and visibility values

The aim of updating them is to make the ant's pheromone strength value be higher when it lies in the better position, and to make its visibility value be higher when its fitness function gradient is larger.

As soon as seek one step for an ant, evaluate its fitness function value and gradient value so that its pheromone strength values and visibility values of current position is updated locally. Let the ant's position in the last step is x_q, its position in the present step is x_r, and its next objective position is x_s. Therefore, local update obeys the following equations.

$$\tau(x_s) = \tau(x_r) + \mathrm{sgn}(\Delta fit) \times |\Delta fit|^{\alpha} \tag{19}$$

$$\Delta fit = fit(x_s) - fit(x_r) \tag{20}$$

$$\eta(x_s) = \eta(x_r) + \mathrm{sgn}(\Delta\partial fit) \times |\Delta\partial fit|^{\beta} \tag{21}$$

$$\Delta\partial fit = \max_{1\le i\le n}\left(\frac{\partial fit}{\partial x_{si}}\right) - \max_{1\le i\le n}\left(\frac{\partial fit}{\partial x_{ri}}\right) \tag{22}$$

When fit isn't a differentiable function, $\Delta\partial fit$ can be calculated by first-order difference

$$\Delta\partial fit = \max_{1\le i\le n}\left(\frac{fit(x_s) - fit(x_r)}{x_{si} - x_{ri}}\right) - \max_{1\le i\le n}\left(\frac{fit(x_r) - fit(x_q)}{x_{ri} - x_{qi}}\right) \tag{23}$$

where $0<\alpha<1$ and $0<\beta<1$.

According to the method mentioned above, all of the ants in colony are operated once, and then use the following equation to update ants' pheromone strength values globally

$$\tau(x_u) = \begin{cases} (1-\rho)\tau(x_u) + \rho \times fit(x_u) & x_u = \tilde{x} \\ (1-\rho)\tau(x_u) & x_u \neq \tilde{x} \end{cases} \tag{24}$$

where $0< \rho <1$, and $\tilde{x}$ is the optimization solution at present.

Ⅲ. PROPOSED NOVEL QUANTUM NEUROCOMPUTING SCHEME

Rotation angle δ_{ij} in hidden layer and weight ω_{jk} of output layer are all the parameters to be adjusted in the network. In this paper, we adopt QAC algorithm introduced in Part B of Section Ⅱ to train these parameters. The detailed steps are shown as follows.

1) Generate the initial colony.
2) Construct QNN.
3) Use training samples to train QNN.
4) We choose the objective position for each ant in the colony according to Eq. (12) and (13). Use quantum rotation gates to implement ants' movement.
5) We map ant's point in the unit space $\tilde{\Omega}$ to the optimization space Ω, and then we calculate the ant's fitness function value and partial derivative.
6) Start iteration, and the iteration time t increases by 1.
7) We update pheromone strength values and visibility values of all the ants in the colony, and then evaluate ant's pheromone strength value.
8) If in some iteration times global pheromone strength value keeps constant, then we implement mutation to the ant which lies in the global optimization position according to Eq. (18) . If the ant's consequent pheromone strength value is better than its previous value, then replace its current position with the global best position.
9) If the iteration time t satisfies t > Max or satisfies its convergence condition, output QNN's the best parameters; otherwise, turn to the step 7).

Fig. 2. Steps of quantum neurocomputing scheme QAC-QNN

We demonstrate the details of some steps in Fig. 2 as

follows.

In step 1), we generate the initial colony randomly according to Eq. (25) . For each ant, assign the same value to its pheromone strength and visibility. We assume that the maximum iteration time is *Max*, and denote that the current time *t* equals 0.

$$a_i = \begin{vmatrix} \cos\varphi_{i1} & \cos\varphi_{i2} & ... & \cos\varphi_{in} \\ \sin\varphi_{i1} & \sin\varphi_{i2} & ... & \sin\varphi_{in} \end{vmatrix} \qquad (25)$$

where $\varphi_{ij} = 2\pi \times Rdm$ in the above equation, and *Rdm* is a random figure in the range (0,1); i=1,2,…,m; j=1,2,…,n; and *m* represents the scale of the ant colony, *n* represents the number of qubits.

In step 2), we denote the corresponding two solution sets about QNN's parameters to be adjusted (rotation angle δ_{ij} and weight of the output layer ω_{jk}) by the amplitudes of the ground state $|0\rangle$ and $|1\rangle$ of the qubit according to each ant's position point.

In step 7), we carry out the following operation to all the ants in colony:

a) We locally update each ant's pheromone strength value and visibility value according to Eq. (19)--(23). After updating that of all ants, we globally update pheromone strength values finally according to Eq. (24) .

b) We evaluate ants' pheromone strength after the last step update. If an ant's pheromone strength value is higher than its value of old optimization position, then we replace its optimization position with its current position. If the ant's a certain old optimization position is better than the acquired current global optimization position, then we replace its global optimization position with that old optimization position.

Ⅳ. APPLICATION EXAMPLE ABOUT PREDICTING PPI

Here, we apply QNN and the scheme QAC-QNN to predict whether there is PPI between two proteins in an organism or not.

This paper uses common yeast dataset, and it is derived from the database *DIP* (http://dip.doe-mbi.ucla.edu/dip). We choose the yeast PPI data which include 5966 couples PPI.

A. Protein Sequence Features Vectorization

In order to implement learning of QNN, we must firstly extract features from protein amino acid sequence and transform its important biological information into QNN's recognizable information.

We adopt *N*-intervals diad protein sequence features extraction method[12]. This extraction method is shown as follows. Firstly, classify amino acids in the protein sequence using a certain physico-chemical property of amino acids. Let they are classified into *M* types. Then find out all the diads with *N* positions interval and take their frequency in the sequence as a feature vector. Here we use *diad* for denoting the combination which are two types chosen arbitrarily from *M* types of amino acids. Obviously, *M*×*M* combination can be achieved. Hence, the feature vector is *M*×*M* dimensions. Without loss of generality, we adopt 1-interval diad in this paper.

Some scholars have found that if two proteins possess similar sequences of hydrophobic amino acids then these two proteins will interact with each other[12]. However, 20 classes amino acids in protein sequence have 20 different hydrophobicity values. If we directly take 20 hydrophobicity values as classification benchmark and adopt 1-interval diad features extraction method mentioned above, each protein sequence will produce feature vectors with 20×20 dimensions. Obviously this will lead to the dimensions disaster problem. Hence, we draw lessons from the literature [12], and divide the 20 amino acids into five classes with *KNN* algorithm according to their hydrophobicity. It is shown as TABLE Ⅰ .

TABLE I. Five classes of amino acids according to their hydrophobicity values

Classes				
1	**2**	**3**	**4**	**5**
K、N、D、E、P、S	R、Q、T、G、A	H、W、Y	F、I、L、M	C、V

Then we adopt 1-interval diad method and choose arbitrarily two classes serial number to combine a diad. We obtain 25 diads such as (1,1) , (1,2) , (1,3) , (1,4) , (1,5) , (2,1) , etc. Then we use such 25 dimensions feature vector to describe each protein sequence.

During study this prediction problem, for these two proteins to predict PPI, we need encode their sequence information firstly and then extract their feature vector. At present, ordinary encoding methods to encode the 2 proteins with PPI are Direct series encoding, Vector plus encoding and Vector minus encoding, etc. In the study of literature [12], among these encoding methods mentioned above, Direct series encoding performs the best in predicting PPI between proteins couple. Therefore, this paper choose the Direct series encoding method to encode the two protein sequences for prediction.

Bring together the 1-interval diad features extraction method mentioned above, we encode this couple of proteins for prediction into 50 dimensions input vector by Direct series encoding method.

B. Constructing QNN and Parameters Optimization

As introduced in the last section, the input sample is a row vector with 50 dimensions. Hence, we set that the number of neurons *n* in the input layer equals 50. With regard to output of QNN, the output result 0 represents that proteins couple for prediction doesn't interact; otherwise, QNN outputs 1. Therefore we use 0 or 1 for denoting output of QNN and then we set that the number of neurons *m* in the output layer equals 1. We set that the number of neurons *p* in the hidden layer equals 15 and the leaning rate α equals 0.8 .

During implementing QAC algorithm to optimize QNN's parameters mentioned above, we set the scale of ant colony to be 100. When the system is initialized, the rotation angle δ_{ij} of hidden layer is assigned a value randomly in the range (- π/2, π/2) , and the weight ω_{jk} of output layer is assigned a value randomly in the range (-1,1) . We assume that the iteration time is 500.

C. Simulation Experiments

1) Indicators to evaluate experiment results

There are merely two types prediction result of proteins couple's PPI merely: one is existing interaction, the other is non-existing interaction. Therefore this problem is a binary classification problem. In biology, for this type of problem, N times cross validation is often used in experiments and test this training method independently[12]. Hence, we adopt 10 times cross validation, and randomly divide all the samples into 10 shares equally. We take 9 shares of samples as the training set and the remaining as the testing set.

Generally, the indicators for evaluating the results of the binary classification are Accuracy (ACC), Sensitivity (SN), Positive Predictive Value (PPV), etc.

2) Results contrast

Here, the method proposed in this paper is denoted by QAC-QNN. For ease of results contrast, we compare QAC-QNN's result for the same data with other methods' result in references. Comparison results are shown as TABLE Ⅱ.

TABLE Ⅱ. Effect comparison of several prediction methods

Methods	SN(%)	PPV(%)	ACC(%)
Martin	63.2	71.5	69.0
Bock	69.8	80.2	75.8
YZGuo	87.3	87.8	87.4
ANN	89.3	86.5	87.9
QAC-QNN	89.8	88.3	88.5

Martin[13] et al. used the same yeast dataset and SVM method to predict. Their experiment ACC was 69.0% . For the same dataset, Bock[14] et al. also used SVM to classify proteins and took advantage of amino acids' physico-chemical property during extracting feature vectors. Their method's ACC reached 75.8% . YZGuo[15] et al. used the same dataset and adopted SVM method to predict. They combined a feature expression through autocovariance and considered the neighbor effects in acid residues sequence so that they reached 87.4% ACC. The literature [12] used traditional artificial neural network (it is denoted by ANN in TABLE Ⅱ) for the same dataset and achieved the prediction ACC 87.9% .

In this paper, our QAC-QNN method uses the quantum-inspired neural network model QNN for the same yeast dataset and combines the quantum ant colony optimization algorithm QAC to optimize QNN's parameters. At last we achieve higher prediction ACC 88.5% .

Ⅴ. CONCLUSIONS

The quantum-inspired neural network QNN is through

searching for rotation angle, and this type of rotation performs round 3 axes of the Bloch sphere. Therefore, in contrast with ANN which is traditional neural network and only searches for every weight singly, QNN can more entirely search in the solution space. In this paper, we also utilize the quantum ant colony optimization algorithm (QAC) to optimize QNN's parameters and propose a novel quantum neurocomputing scheme QAC-QNN which can prevent the solution sinking into local optimization. The simulation experiments prove that the novel quantum neurocomputing scheme is feasible. For predicting PPI of proteins couple, in contrast with SVM and ANN, QAC-QNN achieves better prediction effects.

ACKNOWLEDGMENT

This work is supported by the Fundamental Research Funds for the Central Universities No. 2662018YJ016 and No. 2662017JC029.

REFERENCES

[1] P. E. Bourne, H. Weissig. Structural Bioinformatics. Zhenming Liu (Translation). Beijing: Chemical Industry Press, 2009

[2] Twyman R M. Principles of Proteomics. Henliang Wang (Translation). Beijing: Chemical Industry Press, 2007: pp.65-87

[3] Xiuquan Du. Research on Method of Prediction of Protein-Protein Interaction Using Intelligent Computing. [PhD. thesis] Hefei, Anhui University,2010 (In Chinese)

[4] Lu Cai. Curriculum of Bioinformatics. Beijing: Chemical Industry Press, 2007

[5] Yan C H, Vasant H, Drena D. Predicting Protein-Protein Interaction Sites From Amino Acid Sequence. Technical report, Iowa State University,2002: pp.45-34

[6] Pellegrini, P ;Spada, R , Recensioni - Umanisti bellunesi fra Quattro e Cinquecento - Atti del Convegno, Belluno, 5 novembre 1999. Lettere Italiane.2003, 55(2): pp.293-294

[7] Sprinzak, Ehud. The Lone Gunmen. Foreign Policy.2001, 127: pp.72-73

[8] Martins, A Interpreting from a to b: A Spanish Case Study. Synapse Paris, 2005, 211: pp.22-26

[9] Guo Y.Z, Yu L.Z, Wen Z.N, et.al. 2008. Using support veetor maehine combined with auto covariance to Predict Protein-protein interactions from Protein sequences. Nucleic Acids Research, 36(9): pp.3025-3030

[10] Panchi Li, Guorei Li. Hybrid Quantum-inspired Neural Networks Model and Algorithm. Journal of Electronics & Information Technology. 2016, 38(1): pp.111-118 (In Chinese)

[11] Panchi Li, Shiyong Li. Quantum ant colony algorithm for continuous space optimization. Control Theory & Applications. 2008, 25(2): pp.237-241 (In Chinese)

[12] Juan Cui. Research on Protein-Protein Interactions Based on Integrated Neural Network. [Master. thesis]. Jinan University, 2012 (In Chinese)

[13] Martin S, Roe D, Faulon J L. Predicting protein-protein interactions using signature products. Bioinformatics. 2005, 21(2): pp.218-226

[14] Bock J R, Gough D A. Predicting protein-protein interactions from primary structure. Bioinformatics, 2001, 17(5): pp.455-460

[15] Yanzhi Guo, Lezheng Yu, Zhining Wen, et al. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. Nucleic Acids Research, 2008, 36(9), pp.3025–3030