Sādhanā Vol. 41, No. 11, November 2016, pp. 1299–1309
DOI 10.1007/s12046-016-0561-0

Novel quantum inspired binary neural network algorithm

OM PRAKASH PATEL* and ARUNA TIWARI

Department of Computer Science and Engineering, Indian Institute of Technology Indore, Indore 453552, India
e-mail: oppatel13@gmail.com; artiwari@iiti.ac.in

MS received 16 April 2015; revised 5 April 2016; accepted 5 June 2016

Abstract. In this paper, a quantum based binary neural network algorithm is proposed, named as novel quantum binary neural network algorithm (NQ-BNN). It forms a neural network structure by deciding weights and separability parameter in quantum based manner. Quantum computing concept represents solution probabilistically and gives large search space to find optimal value of required parameters using Gaussian random number generator. The neural network structure forms constructively having three number of layers input layer: hidden layer and output layer. A constructive way of deciding the network eliminates the unnecessary training of neural network. A new parameter that is a quantum separability parameter (QSP) is introduced here, which finds an optimal separability plane to classify input samples. During learning, it searches for an optimal separability plane. This parameter is taken as the threshold of neuron for learning of neural network. This algorithm is tested with three benchmark datasets and produces improved results than existing quantum inspired and other classification approaches.

Keywords. Quantum computing; neural network; quantum gates; classification; separability plane.

1. Introduction

Artificial neural networks have been successfully applied to problems in pattern classification, pattern matching, associative memories, optimization and function approximation [1–3]. Several architectures have been proposed like perceptron, backpropagation, recurrent network, etc. to solve the problem from various fields like mathematics, medicine, economics, computer science and many more [4–6]. The performance of neural network in the mentioned area depends upon several parameters such as network architecture, input data, number of neurons, the number of hidden layers, activation function and weights [7–10]. In this paper, with the help of quantum computing concept and constructive formation of neural network, all these parameters have been optimized. The proposed algorithm uses the quantum computing concept for the selection of weights required to establish the connection at hidden and output layers. Quantum computing concept was, firstly, introduced in classical computing by Narayanan and Moore [11]. The significant work has been done by Han and Kim to solve the knapsack problem using the quantum computing concept with and without termination criteria [12, 13]. Here, qubit q is defined as a smallest unit of information which have better characteristic of the population diversity than other representations. Since qubits are linear superposition of states of probabilistic thus, with the help of Gaussian random generation it gives diversity to select the optimal value of parameters from large subspace. Lu *et al* [2] proposed an algorithm for optimizing artificial neural networks by deciding connection weight and architecture through the quantum computing concept. The quantum computing concept has also been used in several applications. Gandhi *et al* [14] proposed an algorithm to filter EEG signal for brain computer interface. A novel scheme has been proposed by Li and Xu [15] for speech enhancement based on quantum feed-forward neural network, which produces better results than traditional spectral subtraction and Wiener filtering method. Caraiman and Manta [16] proposed an image processing technique using the quantum concept. As network architecture plays an important role in the performance of the system, therefore, selection of an appropriate network architecture is required. There are many algorithms exist which constructively form the network architecture. As [17] proposed an algorithm to design neural network architecture constructively for data classification. Huang and Huang [18] proposed a method to bound number of hidden layer neuron in multilayer perceptron. Similarly, [19] proposed an algorithm to bound on the number of hidden layer neurons for the binary neural network classifier. It offers a high degree of parallelism in the hidden layer formation.

To overcome the issue of finding proper weights and selection of network architecture, recently a quantum inspired binary neural network algorithm is presented by Patel and Tiwari [20, 21]. In this algorithm, the weights of

*For correspondence

the network are decided by quantum computing concept, hence each weight space is decomposed into subspaces in terms of quantum bits. The quantum bits that represent the probability of weight subspaces rather than a specific structure and weight values. Thus using quantum computing the partitioning subspace strategy finds the near-optimal weights. It explores each weight space region-by-region and rapidly finds the promising subspace for further exploitation using Gaussian random generation. The network architecture formed constructively. Here, threshold of a neuron is decided by evaluating 10–15% sample input with weight basis. This method produces good results on benchmark dataset, but deciding the threshold of a neuron may lead to a local optima problem. Therefore, to overcome this issue, in this paper a novel quantum inspired binary neural network algorithm (NQ-BNN) is presented. This paper presented a quantum separability parameter (QSP) or quantum threshold, which help to classify nonlinear separable data into separable data. The selection of quantum threshold or quantum separability parameters using quantum computing concept gives the same advantages as selection of weight through quantum computing. These parameters are updated by the quantum computing concept. Here objective function *sum* has been used to define the quantum states. It means, when the neural network has better objective function in current iteration, the probability of the corresponding connections weights and quantum separability parameters being adopted in the next iteration is increased. If neural network has a bad objective function, then the probability of the corresponding connection weights and quantum separability parameters being adopted in the next iteration is decreased. This proposed approach has been tested on three benchmark dataset, and it is found that it gives better results than existing known methods [17, 20–23].

The paper is organized as follows: section 2 describes preliminaries. Section 3 is presented with proposed methodology. Section 4 is presented with the experimental work and shows the results on three benchmark datasets: breast cancer data, PIMA Indian diabetes and liver disease diagnosis. Section 5 is presented with the concluding remarks.

2. Preliminaries

In this paper a quantum based binary neural network is proposed, which construct a binary neural network using the quantum computing concept. The quantum computing concept has been used to decide the parameters like weights of neural networks and quantum separability parameter (QSP) or threshold. All the parameters have been briefly described throughout this section.

The proposed method forms a neural network structure, which consists of three layers, input layer, hidden layer and output layer. Let $X = (X_1, X_2, X_3, \ldots, X_l)$ denote the input samples, where l is the number of input samples and $X_i = (x_1^l, x_2^l, x_3^l, \ldots, x_n^l)$ where n is the number of attributes in one instance of input sample. Therefore, the number of input layer nodes is equal to n. For jth hidden layer neuron, connection weights are denoted as follows:

$$W_j = \left(w_{j1}, w_{j2}, w_{j3}, \ldots, w_{jn}\right) \tag{1}$$

Here, the number of neurons in the hidden layer is decided constructively. The Output layer contains only one neuron which gives a binary output. In the proposed neural network, the connection weights (W_j) and threshold or quantum separability parameter (λ_j) of neuron are decided using quantum computing concept. The required preliminaries for the quantum computing concept are presented subsequently. The core idea of quantum inspired algorithm is to present solution of a problem in terms of so-called quantum bits Q rather than classical bits. The weight matrix corresponding to Eq. (1) of jth hidden layer neuron in the form of quantum bit Q can be represented as

$$W_j' = \left(Q_{j1}, Q_{j2}, Q_{j3}, \ldots, Q_{jn}\right); \tag{2}$$

The quantum bit (Q_j) can be represented by several qubits (q). The weight W_j' space is decomposed into subspace using quantum bit (Q_j).

$$Q_j = \left(q_{j1}|q_{j2}|\ldots|q_{jk}\right). \tag{3}$$

Here, the k number of qubits which represent quantum bit (Q). A single qubit (q_{ji}) where $i = 1,2\ldots,k$, is the smallest unit of representing information. A qubit is fundamentally different from the binary bit used in traditional digital computers in the sense of representing data. A single binary bit can represent only two states, "0" and "1", whereas the qubit (q_{ji}) has the capability to represent the linear superposition of two states simultaneously, which is determined by probability model [12]. Thus, qubit q_{ji} can be represented as

$$q_{ji} = \alpha_{ji} \mid 0\rangle + \beta_{ji} \mid 1\rangle = \begin{bmatrix} \alpha_{ji} \\ \beta_{ji} \end{bmatrix} \tag{4}$$

where α and β is a complex number representing the probability of qubit in "0" state and in "1" state. A probability model is applied here to decide to qubit in "0" state by α^2 and in "1" state by β^2, where

$$\alpha_{ji}^2 + \beta_{ji}^2 = 1; 0 \le \alpha_{ji} \le 1, 0 \le \beta_{ji} \le 1.$$

As discussed above, a quantum bit (Q_j) formed using qubits $(q_{j1}, (k = 1))$ which represent two states e.g. "0" state or "1" state. An individual quantum bit (Q_j) having two qubits $(q_{j1}|q_{j2}, (k = 1, 2))$ in it represents four states, e.g. "00", "01", "10" and "11". In the same way, three-qubits $((q_{j1}|q_{j2}|q_{j2}), (k = 1, 2, 3))$ system represent eight states,

thus n qubits (q_{jn}, ($k = 1, \ldots, n$)) will have 2^n states. For example, an individual quantum bit Q_j having two qubits can be represented as follows:

$$Q_j = \left\langle \begin{matrix} \alpha_{j1} | \alpha_{j2} \\ \beta_{j1} | \beta_{j2} \end{matrix} \right\rangle. \tag{5}$$

The below example shows representation of the four states of quantum bit (Q_j) having two qubits (q_{j2}).

$$Q_j = (\alpha_{j1} \times \alpha_{j2})\langle 00\rangle + (\alpha_{j1} \times \beta_{j2})\langle 01\rangle + (\beta_{j1} \times \alpha_{j2})\langle 10\rangle + (\beta_{j1} \times \beta_{j2})\langle 11\rangle. \tag{6}$$

It is noted that qubit (q_{ji}) is made of two components α_{ji} and β_{ji}, where each component value lies between "0" and "1". Let us assume that a quantum bit (Q_j) having 2 qubits (q_{j2}) and any random value can be initialized for parameter mentioned in Eq. (4). Here α_{j1}, α_{j2}, β_{j1} and β_{j2} are initialized as follows:

$$Q_j = \left\langle \begin{matrix} 1/\sqrt{2} | 1/\sqrt{2} | 1/\sqrt{2} \\ 1/\sqrt{2} | 1/\sqrt{2} | 1/\sqrt{2} \end{matrix} \right\rangle. \tag{7}$$

With respect to Eq. (5) and Eq. (6), the state representation of quantum bits (Q_j).

$$Q_j = \left(1/2\sqrt{2}\right)\langle 000\rangle + \left(1/2\sqrt{2}\right)\langle 001\rangle + \left(1/2\sqrt{2}\right)\langle 010\rangle + \left(1/2\sqrt{2}\right)\langle 011\rangle + \left(1/2\sqrt{2}\right)\langle 100\rangle + \left(1/2\sqrt{2}\right)\langle 101\rangle + \left(1/2\sqrt{2}\right)\langle 110\rangle + \left(1/2\sqrt{2}\right)\langle 111\rangle. \tag{8}$$

2.1 *Real coded value generation*

The proposed algorithm is inspired by quantum concept which makes use of qubit to represent data. As the classical computer operates on bits and discrete value rather than qubits, therefore, there is the need to convert qubit (q) into real coded value q^{real}. The conversion of real coded value from quantum bits has been done with the help of the observation process.

This process starts by taking random number matrix R_j, where $R_j = [r_{j1} r_{j2} \ldots r_{jk}]$, corresponding to $Q_j = (\alpha_{j1} \mid \alpha_{j2} \mid \ldots \mid \alpha_{jk})$. The value of r_{ji} is selected with the help of random function which generates uniform number between 0 to 1. Then, further mapping is done by using binary matrix S_j where $S_j = [s_{j1} s_{j2} \ldots s_{jk}]$. The value of matrix S_j is generated as follows:

$$\text{if } (r_j \leq (\alpha_{ji})^2) \text{ then } s_j = 1 \text{ else } s_j = 0. \tag{9}$$

To select weights from binary values, the Gaussian random generator has been used with mean value μ and variance σ, represented as $N(\mu, \sigma)$. The observation process shows the process of conversion of a single qubit. As shown in Eq. (4) the qubit (q_{ji}) is formed by using two components α_{ji} and β_{ji}. For processing of qubits, generally α_{ji} component is considered because the value of second component β_{ji} will be $\sqrt{1 - \alpha_{ji}^2}$ [12]. This observation performed for all qubits of Q_j. Thus, this Q_j is utilized for getting real coded value of the weight and separability parameter.

Algorithm 1: *Observation process*

begin
Step 1: q_{ji}, link=0 and random number matrix r_{ji}
  for i=1:k
    q_{ji}=α_{ji}; $0 \leq \alpha_{ji} \leq 1$
    r_{ji}=rand();
    This rand function generate uniform value between 0 and 1.
  endfor
Step 2: **for i=1:k**
    if (r_{ji} <(α_{ji}*α_{ji}))
      s_{ji}=1;
    else
      s_{ji}=0;
    endif
    $link = bin2dec(S_j) + 1$
    if(link~=0)
      $q_{ji}^{real} = N(\mu_{link}, \sigma_{link})$;
    endif
    endfor
end

The observation process can be understood with the help of an example. Let a quantum bit of length two qubits is represented as $Q = \langle 0.707|0.707\rangle$, therefore a random number matrix is generated using random number $R = [0.85\ 0.02]$. Now using Eq. (9) the binary matrix is generated as $S = [01]$. Once the binary matrix is achieved then the formula used to convert a binary number to a decimal value (bin2dec(S)+1) is used here. This return a number between 1 to 4 and corresponding four Gaussian random values are also mentioned for example N(0.25, 0.03), N(0.40, 0.03), N(0.55, 0.03), and N(0.70, 0.03). As a binary value achieved here return 2 as decimal value, therefore the real coded value corresponding to quantum bit Q is selected from N(0.40, 0.03).

2.2 *Qubit updation*

This observation process helps to convert qubits W' into real coded W matrix for g iteration. However, the proposed method finds the best value of weight from the large subspace provided by the quantum concept. The appropriate value of weight can find out during several iterations. Therefore, there is a need of updating weight values. The new value of weight (W) is generated through quantum bits W' with quantum update function. To update W'_{g+1} from W'_g quantum rotation gates as required, which is described as follows:

$$U(\Delta\theta) = \begin{vmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{vmatrix} \tag{10}$$

where $\Delta\theta$ is a rotation angle which use to generate W'_{g+1} from W'_g. In the proposed algorithm, the length of individual qubit $Q_i^g = (\alpha_{i,1}^g \mid \alpha_{i,2}^g \mid \ldots \mid \alpha_{i,k}^g)$ is considered as 2 ($k = 2$). It means that W will be selected from four subspaces.

$$\begin{vmatrix} \alpha_i^{g+1} \\ \beta_i^{g+1} \end{vmatrix} = \begin{vmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{vmatrix} * \begin{vmatrix} \alpha_i^g \\ \beta_i^g \end{vmatrix}. \tag{11}$$

As presented in Lu *et al* [2], $\Delta\theta$ is calculated on the basis of the objective function. sum^* and sum_g are the objective function values which depend on parameters count1 and count2 showing the number of learnt samples in the proposed algorithm discussed in the next section. This objective function gives better exploitation in the next iteration. The objective function parameters are sum^* represent as the best objective function value in g iterations and sum_g objective function value of current iteration. sum^* and sum_g are evaluated based on the number of samples learnt. As shown in the observation process, each qubits α_{ji} is associated with binary value s_{ji}, therefore, a mapping is done between sum^* and binary string s to update individual qubit. If objective function values sum_g corresponding to s is worse than that of objective function value in sum^*, and state of s is zero and best objective function state of s is one, then decrement in probability of α_{ji} may produce the worst result. Therefore, to increase the probability of α_{ji} to one, $\Delta\theta$ made negative. While, if the objective function values sum_g corresponding to s is better than the objective function in sum^*, and state of s is one and best objective function state is zero, then increasing probability of α_{ji} to one, may produce the worst result. Therefore, to update α_{ji}, angular displacement made positive $\Delta\theta$. In other cases, angular displacement will remain zero. The value of angular displacement must be selected in such a way so that it can cover the maximum value of α in the range of (0 1) and also should not take many iterations to cover these values. Therefore, $\Delta\theta$ must be initialized between $(0.01 \times \pi, 0.05 \times \pi)$ [2]. The evaluation of objective function function is done in the proposed algorithm that is discussed next.

For preventing the quantum bit α_i^g from acquiring values 0 or 1, following constraints are applied:

$$\alpha_i^g = \begin{cases} \sqrt{\epsilon}, & \text{if } \alpha_i^g < \sqrt{\epsilon} \\ \alpha_i^g & \text{if } \sqrt{\epsilon} \le \alpha_i^g \le \sqrt{1-\epsilon} \\ \sqrt{1-\epsilon} & \text{if } \alpha_i^g > \sqrt{1-\epsilon} \end{cases} \tag{12}$$

where the value that assigned to ϵ is very small (approximately approaching to zero), so that it can cover maximum value in the range of (0, 1). Based on these basic concepts, a novel quantum binary neural network learning algorithm (NQ-BNN) algorithm is designed, which forms three layer network structure. Network structure has input layer, hidden layer and output layer. This works on binary form of input, and also generates binary form of output. It works for two-class classification problem.

3. Proposed approach

In this section, a novel quantum binary neural network learning algorithm (NQ-BNN) is proposed, which makes use of a novel quantum separability parameter or threshold. The quantum computing concept helps in evolving process of connection weight (W_g) and separability parameter (λ). First, the basic principle of using quantum bits in learning is discussed, then the quantum separability parameter is proposed. Finally, learning algorithm NQ-BNN is modelled by making use of this quantum separability parameter.

3.1 *Basic principle*

The proposed algorithm constructs neural network architecture by using the quantum computing concept. It forms a three-layer network structure having input, hidden and output layer. Here, $X = (X_1, X_2, X_3, \ldots, X_{c_1})$ and $Y = (Y_1, Y_2, Y_3, \ldots, Y_{c_2})$ denotes the instance of input samples of two different classes, which is present in the binary form.

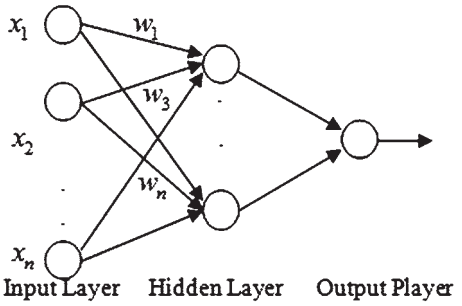

Figure 1. Basic architecture.

Let, X_i and Y_i are one of the instances of class A and class B respectively. Each instance of $X_i = (x_1, x_2, x_3, \ldots, x_n)$ and $Y_i = (y_1, y_2, y_3, \ldots, y_n)$ has n attributes. As each instance has n attributes, therefore input nodes will be equal to n. The number of neurons in the hidden layer is decided constructively. As proposed system deals with two-class problem, therefore only one neuron required at the output layer. Figure 1 shows the basic architecture of the proposed system.

In the proposed learning algorithm, we make use of step function as an activation function for forming the neuron which is given as follows.

$$\text{net}_j = \sum_{i=1}^{n} W_j \times X_i \quad \textbf{or} \quad \sum_{i=1}^{n} W_j \times Y_i \tag{13}$$

$$f(\text{net}_j) = \begin{cases} 1 & \text{if} \quad \text{net}_j \leq \lambda_j \\ 0 & \text{if} \quad \text{net}_j > \lambda_j. \end{cases} \tag{14}$$

Training starts by taking a single neuron in hidden layer first. The weights of this neuron is initialized as $W'_g = (Q^g_{w1}, Q^g_{w2}, Q^g_{w3}, \ldots, Q^g_{wn})$, where $Q_{wi} = (0.707 \mid 0.707)$, with $k = 2$ and $g = 1$. Here K is a subspace selection of weights, and g (user defined variable) is the maximum number of iterations to update quantum weights to get the optimized result. After initialization of quantum bits, vector R is generated with random values. Now the observation process starts. It will generate vector S^g_{wi} in terms of bits. With the help of S^g_{wi} and Gaussian random value generator, weight matrix is finalized for first iteration $g = 1$. Then all the samples of class A and class B are applied to the neuron along with weight W_g (real coded number with respect to W'_g). Thus, after finalizing a neuron, it is checked against samples of both the classes A and B. For proper separation of samples of different class quantum separability parameter is proposed next.

3.2 *Quantum separability parameter (QSP) (λ')*

To produce better separating plane for two classes, quantum separability parameter or quantum threshold is proposed. This separability parameter (λ'_t) is initialized, which takes a quantum value denoted as

$$\lambda'_t = (\alpha_i \mid \alpha_{i+1}); \tag{15}$$

The value is set to (0.358|0.586) for selecting the value from four subspaces. Quantum separability parameter is updated for the time t (user defined variable) in each iteration (g). After initialization of quantum bits for separability parameter (λ'_t), the observation process is used. The observation process generates a random number matrix R. This random number and quantum value with the help of step 2 of observation process function, generates binary value matrix S_t. Now, the real coded value of λ_t (corresponding to quantum value λ'_t) is achieved with the help of matrix S_t and Gaussian random generator. The value of quantum separability parameter is finalized by exploring in 2^k subspace thus introducing diversity to reach the optimal value. The real coded value λ_t is compared with net_A and net_B at $g = 1$ to find out the optimal separability plane for two class data. This λ_t quantum separability parameter is updated for t iteration for weight value W_g at iteration $g = 1$. After completion of all iterations t, best value of λ_t is selected corresponding to the best value of the objective function. Now quantum weights are updated for iteration $g = 2$ and again the same process is implemented to find out best value of λ_t in all iterations of t. The process will continue for the iteration g to find out best value of weights W and quantum separability parameter λ. To update quantum weights W'_g and quantum separability parameter (λ'_t), Eq. (10), Eq. (11) and Eq. (12) along with quantum updation process is used. The overall updation process is presented in the form of a pseudo-code as quantum update function.

3.3 *Design of NQ-BNN*

The NQ-BNN is designed to classify two-class problem using quantum computing concept. The quantum bits are used to evaluate weights of network termed as quantum weights. These quantum weights are evaluated using an observation process with the help of the Gaussian random number generator. To classify an input sample more accurately, a novel concept is proposed termed as quantum separability parameter. Furthermore, the designed algorithm constructively form the network and work only for binary data. In this algorithm, some necessary parameters have been used, which is described as follows: X_i and Y_j are an input sample of class A and class B respectively for the learning process. Two values *count*1 and *count*2 have been taken, which describe the number of samples of class A and B that are learnt. Here S^* vector is used which stores best value from S^g corresponding to the best result or value of the sum denoted by sum^*. The overall process is explained in the form of an algorithm which is presented next:

Algorithm 2: *NQ-BNN*

Step 1: Take Input sample as $(X_1, X_2, X_3, ..., X_{c_1})$ and $(Y_1, Y_2, Y_3, ..., Y_{c_2})$
Take first neuron at hidden layer and initialize with the weights $W_g^{'}$ in terms of quantum bits

$$W_g^{'} = (Q_{w1}^g, Q_{w2}^g, Q_{w3}^g, \dots, Q_{wn}^g)$$

where $g = 1, ..., m$; m is the number of iterations to update weights
$sum^* = 0$
$S^* = 0$
Step 2: **for g=1 to m**
Call observation process($W_g^{'}$) (For each Q_{wi})
Call Quantum Separability Parameter(W_g)
if($sum_g^* \geq (c_1 + c_2)$)
Stop learning
else
sum^*=max(sum^*, sum_g^*)
Evaluate sum_g^*, sum^*, $s_k^{g,i}$, $s_k^{*,i}$
and update quantum bits for evolved quantum weight $W_{g+1}^{'}$
by using (sum_g^*, sum^*), Eqs. (10)–(12) and quantum update Table I for the same neuron
endif
endfor
g=g+1
Step 3: **if** (($g == m$) $\wedge$ ($sum_g^* \leq (c_1 + c_2)$))
Add new neuron for unlearnt sample (($c_1 + c_2$)-sum^*) and finalize its weight by using Step-1 and Step-2
endif

Quantum separability parameter(W_g)

Step 1: Initialization of different parameters
for t=1 to z
z is user defined variable to update $\lambda^{'}$
$\lambda_t^{'} = (\alpha_i \mid \alpha_{i+1})$;
count1=0;
count2=0;
$sum_\lambda^* = 0$;
Call observation process($\lambda_t^{'}$) to generate discrete value λ_t from quantum value $\lambda_t^{'}$
for i=1 to c_1
$net_A(i) = \sum W_g \times X_i$
if($net_A(i) > \lambda_t$)
increase count1 by 1;
endif
endfor
for j=1 to c_2
$net_B(j) = \sum W_g \times Y_j$
if($net_B(j) \leq \lambda_t$)
increase count2 by 1;
endif
endfor
($sum^t = count1 + count2$);
sum_λ^*=max(sum_λ^*, sum^t)
update quantum bits for $\lambda_{t+1}^{'}$ by using (sum^t, sum_λ^*), Eqs. (10)–(12) and quantum update table 1
generate updated real coded value of λ_{t+1} corresponding to $\lambda_{t+1}^{'}$ by using observation process
endfor
$t = t + 1$
sum_g^*=sum_λ^*
return sum_g^*;

Note: The observation process is common to both, in generating real coded value of weight (W_g) and quantum separability (λ_t) parameters.

Note: The observation process is common to both, in generating real coded value of weight (W_g) and quantum separability (λ_t) parameters.

The above algorithms use quantum update function to generate new quantum value of weights W and quantum separability parameter λ. Table 1 is shown for updation of qubits for weights, the same process will be implemented to update qubits of separability parameter (λ_t) by using objective function as sum^t, sum_λ^* and binary bits accordingly.

Table 1. Qubits updation.

s_{ji}^{g}	s_{ji}^{*}	$sum_{g}^{*} < sum^{*}$	$\Delta\theta$
0	0	False	0
0	0	True	0
0	1	False	$-0.03 * \Pi$
0	1	True	0
1	0	False	$0.03 * \Pi$
1	0	True	0
1	1	False	0
1	1	True	0

3.4 *Illustration of NQ-BNN*

The illustration of (algorithm 2) NQ-BNN with quantum separability parameter (QSP) is done with the help of one small numerical example as follows:

Let us consider a dataset consist of 16 samples, where each sample having four features (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111). For class A, the set of samples are divided into training and testing set. Let us take training samples for class A as (0000, 0001, 0010, 0011, 0100, 0101, 0110) and only one test sample as (0111). Similarly for class B set of samples are divided, the training samples are (1001, 1010, 1011, 1100, 1101, 1110, 1111) and testing sample is (1000). The proposed NQ-BNN with QSP is illustrated with the help of following steps.

Step 1: To start designing hidden layer first we take a neuron and initialize its quantum weights in terms of quantum bits as $W_1' = (0.352|0.725, 0.907|0.854, 0.524|0.542, 0.924|0.222)$. Also initialize parameters like number of iterations to update quantum weights (g), objective function sum^{*} and binary value S^{*} as $g = 1:10$, $sum^{*} = 0$ and $S^{*} = 0$.

Step 2: For first iteration $g = 1$, observation process (algorithm 1) is called to convert quantum weights W_1' into real coded value matrix W_1. This is being done by first generating random number matrix as R = (0.5|0.32, 0.89|0.25, 0.37|0.97, 0.39|0.95) corresponding to each qubit (q_{ji}) of quantum weights W_1'. With the random number matrix R, quantum weight matrix W_1' and Eq. (9), the binary bit matrix can be formed as $S = (11, 01, 10, 10)$. With the help of matrix S and Gaussian random number generator, the real coded value can be formed as $W_1 = (0.45, 0.25, -0.63, 0.73)$. Now this real coded W_1 is applied into quantum separability function.

Step 3: In quantum separability function, first initialize parameters, the number of iteration to update QSP is $t = 1{:}20$, count1 = 0; count2 = 0; objective function parameter of quantum separability parameter $sum_{\lambda}^{*} = 0$; and quantum separability parameter in terms of qubits as $\lambda_1' = (0.907|0.562)$. This, qubit representation of λ_1', is then converted to real coded value by applying observation process. Let the real coded value achieved as $\lambda_1' = (0.52)$.

Note: In the process of converting quantum weights and quantum separability parameter into real coded value, there is need to keep track of binary bits represented by S for each qubit. These binary bits are required at the time of updating qubit as mentioned in qubit updation process and table 1. This updation of qubit is further explored and resumed.

Step 4: After getting real coded value weight W_1 and separability parameter λ_1, the QSP is further proceed to evaluate the number of samples learnt as net_A and net_B and compared with separability parameter. Accordingly, value of $count1$, $count2$ and sum^{1} $(t = 1)$ is calculated. After this the sum_1 is compared with sum_{λ}^{*} and maximum value is assigned to sum_{λ}^{*}. After assigning the maximum value to sum_{λ}^{*}, qubits are updated. Thus with the help of λ_1', λ_2' is evaluated.

Step 5: The process in step 4 is repeated to keep on evolving λ_t' for 20 iterations i.e. $t = 1{:}20$. The final value of λ_t is selected for which maximum number of samples are learnt or for the best value objective function sum_{λ}^{*} is achieved. The best objective function maximum value corresponding to separability parameter is assigned to sum_{g}^{*}. Let us say out of 14 number of samples, total eight number of samples are learnt by using best value of separability parameter among $\lambda_1, \ldots, \lambda_{20}$ and weight value W_1. Therefore the process will continue until network is learnt for all 14 number of samples or until all iterations (t) are over, to update λ_t'.

Step 6: If all the samples are not learnt then, for the next iteration $(g = 2)$ for the same neuron, quantum weight value W_2' is evolved. By repeating step number 5, with the help of real coded value of weight W_2, λ_t' is evaluated. The quantum weight W_g' evolvement process will be continued till $G = 1{:}10$ iterations or neural network learn for all training set samples. Thus, after 10 iterations $G = 1{:}10$, if all samples are not learnt then add a new neuron and follow step 1 to step 6.

Step 7: The process of adding new neuron would be continued till all samples are learnt or in two successive neurons, number of unlearnt samples remains same.

4. Experimental results

The proposed NQ-BNN algorithm is tested on three benchmark datasets: breast cancer dataset, PIMA Indian diabetes dataset and BUPA liver dataset, which is taken from UCI Machine learning repository [24].

4.1 *Experimental setup*

The experiment is carried on Intel core, I-5 processor with 4 GB RAM on windows 7 operating system. Initially, for the given input dataset quantum weights W_g' are initialized

Table 2. Class distribution of breast cancer dataset.

Index	Class name	Class size	Class distribution (%)
C1	Negative	458	65.52
C2	Positive	241	34.47

with quantum values. The qubits of the quantum separability parameter (λ_t') are also initialized as 0.707|0.707. Using Eqs. (10)–(12) and qubit updation procedure the quantum separability parameter and quantum weights are updated, where displacement angle ($\Delta\theta$) has been used as $0.03 * \Pi$. During this update process, it is insured that quantum weights should not converge to "0" and "1". Therefore, the limiting parameter ϵ has been taken as 0.001. The proposed approach is tested using tenfold cross validation scheme. As the proposed algorithm takes input in terms of binary bits, therefore, real value is converted in terms of binary bits. The number of binary bits to represent each data is decided by observing highest numerical value in dataset.

4.1a *Classification of breast cancer dataset:* The breast cancer dataset has total 699 instances and 9 attributes in terms of real value. The dataset has converted into binary values for applying on the proposed algorithm. Every sample consists of 9 values corresponding to 9 attributes. For converting into binary form, each value would be represented by 3 bits according to maximum value in dataset. Thus every sample is converted into 27 bits and these binary values of every samples are taken as input to NQ-BNN. The description of breast cancer dataset has been given in table 2. In this experiment, dataset have been splitted into tenfolds, where 629 instances (ninefolds) have been used for the training purpose, and 70 instances (onefold) are used for the testing purpose. The training of neural network with cancer dataset produces best results with two numbers of hidden layer neurons. The separability value that achieved from the first and the second neuron is 0.52 and 0.36 respectively. The best results have been achieved in $g = 75$ number of iterations. Table 3 shows the classification result in terms of various parameters as training accuracy, generalization accuracy and number of neurons in the hidden layer. As it can be seen that NQ-BNN achieves good training and generalization accuracy with only two neurons in the hidden layer. Table 3 shows the best training accuracy as 98.6354% and worst training accuracy 95.0325%. The best generalization accuracy is 99.9586% and worst generalization accuracy is 98.6584%. The average training accuracy and generalization accuracy is 97.0816% and 99.6501% respectively. Table 4 shows the comparison of the proposed approach with a novel discretization technique using class attribute interval average and Q-BNN in terms of generalization accuracy [20, 22]. Table 4 shows that the proposed approach produces generalization accuracy as 99.6501%, which is far better than other available methods.

Table 3. Results of NQ-BNN for breast cancer dataset.

	Breast cancer dataset		
	Hidden layer neuron	Training accuracy	Testing_accuracy
Set 1	2	95.0325	99.9574
Set 2	2	97.1854	99.8415
Set 3	2	95.2587	99.9558
Set 4	2	97.0325	99.9586
Set 5	2	98.6354	99.8248
Set 6	2	97.2576	99.8124
Set 7	2	98.1254	98.6584
Set 8	2	97.0221	99.5876
Set 9	2	98.0124	99.9477
Set 10	2	97.2546	98.9568
Average		97.08166	99.6501

Table 4. Comparison of various learning algorithms with NQ-BNN on breast cancer dataset.

Generalization accuracy (%)				
Methods	MLP classifier	Naive bayes classifier	Decision tree classifier	Radial basis function classifier
NQ-BNN*	**99.6501**			
Q-BNN*	**99.63**			
EW	94.57	**97.28**	91.3	95
EF	94.71	96.28	90.8	95
ChiMerge	92.89	91.88	93	92
IEM	74.69	81.68	93.6	80.98
CAIM	93.56	93.99	93.8	93.42
CACC	95.14	95.28	94.1	94.85
CAIA	**95.42**	96.57	**96.57**	**95.99**

Bold numerical values show the best accuracies for particular classifiers like MLP classifier, Naive bays classifier, Decision tree classifier, and Radial basis function classifier. The other bold value shows the proposed results for NQ-BNN

*NQ-BNN and Q-BNN does not use any inbuilt classifier

4.1b *Classification of PIMA Indian diabetes dataset:* The PIMA Indian diabetes dataset has 768 instances and every sample has 27 real values corresponding to each 27 attributes. In the same way as done for cancer dataset each real value is represented by 3 bits. Thus every sample is converted into 81 bits. The description of the PIMA Indian diabetes dataset has been given in table 5. For the training and the testing purpose, the dataset has been splitted into tenfold, where 692 instances (ninefold) have been used for training and 76 instances (onefold) have been used for the testing purpose. In the training phase, the best results are achieved with three numbers of hidden layer neurons. The separability values of hidden layer neurons are achieved as 0.45, 0.86. and 0.66 respectively. The neural network got

Table 7. Comparison of various learning algorithms with NQ-BNN on PIMA Indian dataset.

Methods	Training accuracy (%)	Generalization accuracy (%)
NQ-BNN	**92.74853**	**88.28657**
Q-BNN	85.3	85.6
MTiling-real	85.3	80.6
MPyramid-real	81.3	80.3
Perceptron	81.2	80.9

Bold values show the best result corresponding to the proposed approach

Table 8. Class distribution of BUPA liver dataset.

Index	Class name	Class size	Class distribution (%)
C1	Negative	198	58.23
C2	Positive	142	41.76

Table 9. Results of NQ-BNN for BUPA liver dataset.

	BUPA liver dataset		
	Hidden layer neuron	Training accuracy	Testing_accuracy
Set 1	3	91.5524	95.34
Set 2	3	91.2457	95.2485
Set 3	3	91.3658	95.2475
Set 4	3	90.4578	95.6845
Set 5	3	92.1247	94.6235
Set 6	3	91.4583	94.7851
Set 7	3	91.7452	94.2519
Set 8	3	91.4578	95.2156
Set 9	3	91.2457	94.3615
Set 10	3	90.1458	93.4715
Average		91.27992	94.82296

Table 10. Comparison of various learning algorithms with NQ-BNN on BUPA liver dataset.

Classification algorithm	Accuracy (%)	Precision
NQ-BNN	**94.82296**	**95.35**
QBNN-L	90.35	94.00
Logistic	67.39	75.00
Linear logistic regression	69.57	74.70
Gaussian processes	73.91	79.01
Logistic model trees	68.12	73.49
Multilayer perceptron	68.84	76.32
K-STAR	59.42	71.43
RIPPER	64.49	71.25
Neural net	73.91	77.65
Rule induction	64.49	76.56
Support vector machine	69.23	75.00
Classification and regression trees	66.35	77.36

Bold values show the best result corresponding to the proposed approach

small value of $\Delta\theta$ may increase the number of iterations. The appropriate value of $\Delta\theta$ can be selected from the range of $\Delta\theta \in (0.01 \times \pi, 0.05 \times \pi)$. Therefore, to make a tradeoff between the maximum number of values (which is between "1" and "0") and number of iterations for the faster convergence of a problem, the angular displacement $\Delta\theta$ has been selected as $0.03 \times \pi$ [2, 12].

The quantum based updation process performs well, even with a small population, without premature convergence as compared to the conventional genetic algorithm [12]. As weight selection process using quantum computing concept is compared to another method as fuzzy neural network, in which weights are generally center point of the input dataset. In fuzzy, the parameter selection process is done through random selection, which may or may not belong to dataset point. Due to which algorithm may converge to local maxima and minima problem, whereas quantum approaches gives better search space with the help of Gaussian random generator.

5. Conclusion

This paper presents a novel quantum inspired binary neural network algorithm. This algorithm forms a neural network constructively by using the quantum computing concept for deciding connection weights and separability parameter. The proposed quantum separability parameter helps to find out the proper plane to classify the input data. The NQ-BNN algorithm automatically forms an optimal network structure with the above-mentioned parameters. The algorithm has been tested on the breast cancer dataset, PIMA Indian diabetes dataset and liver disease diagnosis dataset. The results show improvement in classification accuracy when compared to the methods like [17, 20–23].

References

[1] Gupta A K and Singh Y P 2011 Analysis of bidirectional associative memory of neural network method in the string recognition. In: *Proceeding of 2011 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp 172–176, IEEE

[2] Lu T-C, Yu G-R and Juang J-C 2013 Quantum-based algorithm for optimizing artificial neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 24(8): 1266–1278

[3] Wang T and Wang Y 2010 Pattern classification with ordered features using MRMR and neural networks. In: *Proceeding of 2010 International Conference on Information, Networking and Automation (ICINA),* pp 2128–2131, IEEE

[4] Aydin M and Celik E 2013a Assamese character recognition with artificial neural networks. In: *Proceeding of 2013 21st International Conference on Signal Processing and Communications Applications Conference (SIU)*, pp 1–4, IEEE

[5] Aydin M and Celik E 2013b Assamese character recognition with artificial neural networks. In: *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, pp 1–4, IEEE

[6] Turnip A, Hong K-S and Ge S S 2010 Backpropagation neural networks training for single trial eeg classification. In: *Proceeding of 2010 29th International Conference on Chinese Control Conference (CCC),* pp 2462–2467, IEEE

[7] Chan L-H, Salleh S-H and Ting C-M 2009 Pca, lda and neural network for face identification. In: *Proceeding of 2009 4th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp 1256–1259, IEEE

[8] Ou G and Murphey Y L 2007 Multi-class pattern classification using neural networks. *Pattern Recognit.* 40(1): 4–18

[9] Sarikaya R, Hinton G E and Deoras A (2014) Application of deep belief networks for natural language understanding. *IEEE/ACM Trans. Audio Speech Lang. Process.* 22(4): 778–784

[10] Sun B 2013 Analysis and detection of nonlinear analogue based on variable threshold value neuron. In: *Proceedings of 2013 Fifth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA),* pp 225–228, IEEE

[11] Narayanan A and Moore M 1996 Quantum-inspired genetic algorithms. In: *Proceeding of 1996 International Conference on Evolutionary Computation,* pp 141–144, IEEE

[12] Han K-H and Kim J-H 2002 Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Trans. Evolut. Comput.,* 6(6): 580–593

[13] Han K-H and Kim J-H 2004 Quantum-inspired evolutionary algorithms with a new termination criterion, H_ε gate, and two-phase scheme. *IEEE Trans. Evolut. Comput.* 8(2): 156–169

[14] Gandhi V, Prasad G, Coyle D, Behera L and McGinnity T M 2013 Quantum neural network-based eeg filtering for a brain-computer interface. *IEEE Trans. Neural Netw. Learn. Syst.* 25(2): 278–288

[15] Li F and Xu G 2009 A novel scheme of speech enhancement based on quantum neural network. In: *Proceeding of 2009 International Asia Symposium on Intelligent Interaction and Affective Computing (ASIA)*, pp 141–144, IEEE

[16] Caraiman S and Manta V 2012 Image processing using quantum computing. In: *Proceeding of 2012 16th International Conference on System Theory, Control and Computing (ICSTCC)*, pp 1–6, IEEE

[17] Parekh R, Yang J and Honavar V 2000 Constructive neural-network learning algorithms for pattern classification. *IEEE Trans. Neural Netw.* 11(1): 436–451

[18] Huang S-C and Huang S-C 1991 Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Trans. Neural Netw.* 2(1): 47–55

[19] Chaudhari N S and Tiwari A 2010 Binary neural network classifier and it's bound for the number of hidden layer neurons. In: *Proceeding of 2009 Control 11th International Conference on Automation Robotics & Vision (ICARCV)*, pp 2012–2017, IEEE

[20] Patel O P and Tiwari A 2014 Quantum inspired binary neural network algorithm. In: *Proceeding of 2014 International Conference on Information Technology (ICIT),* pp 270–274, IEEE

[21] Patel O P and Tiwari A 2015 Liver disease diagnosis using quantum-based binary neural network learning algorithm. In: *Proceedings of 2015 Fourth International Conference on Soft Computing for Problem Solving*, pp 421–430, Springer

[22] Abdulloh Baka S V and Wiphada Wettayaprasit 2014 A novel discretization technique using class attribute interval average. In: *Proceeding of 2014 Fourth International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, pp 95–100, IEEE

[23] Bahramirad S, Mustapha A and Eshraghi M 2013 Classification of liver disease diagnosis: Acomparative study. In *Proceeding of 2013 Second International Conference on Informatics and Applications (ICIA)*, pp 42–46, IEEE

[24] Lichman M 2013 *UCI machine learning repository.*http://archive.ics.uci.edu/ml

[25] Islam M M, Yao X and Murase K 2003 A constructive algorithm for training cooperative neural network ensembles. *IEEE Trans. Neural Netw.* 14(4): 820–834