Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9319364, 12 pages
https://doi.org/10.1155/2020/9319364

Research Article

Construction and Optimization of Fuzzy Rule-Based Classifier with a Swarm Intelligent Algorithm

Li Mao, Qidong Chen, and Jun Sun

Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, Wuxi, Jiangsu 2141222, China

Correspondence should be addressed to Jun Sun; sunjun_wx@hotmail.com

Received 18 September 2019; Revised 10 March 2020; Accepted 17 March 2020; Published 27 April 2020

Academic Editor: Jose A. Lozano-Galant

In this paper, we propose a particle swarm optimization method incorporating quantum qubit operation to construct and optimize fuzzy rule-based classifiers. The proposed algorithm, denoted as QiQPSO, is inspired by the quantum computing principles. It employs quantum rotation gates to update the probability of each qubit with the corresponding quantum angle updating according to the update equation of the quantum-behaved particle swarm optimization (QPSO). After description of the principle of QiQPSO, we show how to apply QiQPSO to establish a fuzzy classifier through two procedures. The QiQPSO algorithm is first used to construct the initial fuzzy classification system based on the sample data and the grid method of partitioning the feature space, and then the fuzzy rule base of the initial fuzzy classifier is optimized further by QiQPSO in order to reduce the number of the fuzzy rules and thus improve its interpretability. In order to verify the effectiveness of the proposed method, QiQPSO is tested on various real-world classification problems. The experimental results show that the QiQPSO is able to effectively select feature variables and fuzzy rules of the fuzzy classifiers with high classification accuracies. The performance comparison with other methods also shows that the fuzzy classifier optimized by QiQPSO has higher interpretability as well as comparable or even better classification accuracies.

1. Introduction

Pattern classification, as one of the most important problems in the field of pattern recognition and machine learning, attempts to assign each input value to one of a given set of classes [1]. It involves design of a classifier and the use of the classifier to classify the given data. Generally, in practical classification problems, uncertainty or noise interference is always contained in the data, making it difficult to classify the data accurately with classical crisp classification methods [2]. In contrast, fuzzy sets and fuzzy logic provide a means for soft classification of data that allows us to analyze the data samples in a more sensitive way. For example, the use of fuzzy rules for classification is known to be a good method for classification of knowledge representation, which is similar to human knowledge expression, so that it leads to efficient, transparent, and easily interpretable fuzzy classifiers [3]. During the past few decades, fuzzy classification has been widely used in many fields, including speech recognition, character recognition, text classification, image processing, and weather forecast, to name only a few [4–9].

Fuzzy rule-based classification for a practical problem employs two kinds of methods of extracting fuzzy rules [10]. The first one is to generate fuzzy inference rules based on the professional knowledge of experts in a particular field, while the second one is to reason out corresponding fuzzy IF-THEN rules from a set of given data [9]. In this paper and other related literature, the so-called fuzzy classification employs the second type of method to generate a set of fuzzy classification rules. For the second type of methods, researchers also have proposed different approaches for the purpose of effectively extracting the fuzzy classification rules. For example, Abe and Lan [11] proposed to carry out clustering on the sample data first and then define a fuzzy rule for each cluster. Castellano and Fanelli [12] designed a grid partition method to generate the initial fuzzy classifier

and then used the artificial neural network to reduce the number of the fuzzy rules. An important class of approaches in this type is to employ a training algorithm to perform the training task or to optimize fuzzy classification rules [13]. Many heuristic methods, such as genetic algorithm (GA), simulated annealing, genetic programming, and particle swarm optimization (PSO), have been used to optimize the fuzzy rules of classifiers [14–21]. These methods act as the training algorithms to construct or optimize the fuzzy rule base to maximize the classification accuracy and minimize the number of the fuzzy rules on the given training samples.

The goal of this paper is to develop a novel effective method to optimize the fuzzy classifier based on a modified version of quantum-behaved particle swarm optimization (QPSO), which incorporates quantum computing principles into the original QPSO algorithm. As a variant of PSO, the QPSO algorithm has been shown to successfully solve a wide range of optimization problems due to its stronger search ability [22–27]. However, since the QPSO was designed aiming for continuous optimization problems, it cannot be directly applied to optimization of fuzzy classification rules, which is known as a discrete optimization problem. Hence, in this paper, we incorporate a quantum bit operation into the QPSO algorithm and discretize the algorithm to be suitable for optimizing fuzzy rule-based classifiers. Quantum computing principles have been employed to design evolutionary algorithms by many researchers [28–33]. For example, Han and Kim [30] proposed the quantum-inspired evolutionary algorithm (QEA), which is characterized by the concepts of qubits (Q-bit) and superposition of states. Hammed et al. [33] designed a quantum-inspired particle swarm optimization in which the rotation angle in the rotation gate is represented by the velocity of the particle. The proposed algorithm is motivated by these works and is denoted as quantum-inspired QPSO (QiQPSO), in which each individual particle is encoded as a string of qubits and is updated by a rotation gate with the rotation angle evolving according to the iterative equation of the original QPSO.

The paper is structured as follows. In Section 2, we describe how to construct a fuzzy rule-based classification system. Section 3 presents the proposed QiQPSO, and Section 4 describes the procedure of how to construct an initial fuzzy rule-based classifier and optimize its fuzzy rule base further by using the QiQPSO algorithm. Experiment results are presented and analyzed in Section 5. Finally, the paper is concluded in Section 6.

2. Fuzzy Rule-Based Classifiers

Fuzzy rule-based classifier is an important application of fuzzy sets and is essentially a set of IF-THEN classification rules [2]. The real-world classification problems generally contain uncertainty noise interference, which can be tackled effectively by fuzzy rule-based classification methods. The fuzzy rule-based classification system has many advantages such as its consistence with representation of human knowledge, better classification performance, understandability, and strong readability and interpretability.

There are three types of fuzzy classifiers corresponding to three different types of consequents of the fuzzy rules [2]. The first type of consequents only indicates the class that the input data belongs to. That is, it has the following form:

$$R_k\colon\ \ldots, \text{THEN class is } y_{o(k)}, \tag{1}$$

where y_(o(k)) denotes the class represented by the o(k)-th input. The second type of consequents of fuzzy rules is expressed by

$$R_k\colon\ \ldots, \text{THEN class is } y_{o(k)} \text{ with } CF_{k,o(k)}, \tag{2}$$

where $y_{o(k)}$ denotes the class that the $o(k)$-th input data belongs to and $CF_{k,o(k)} \in [0,1]$ refers to the degree of confidence with which class the input data belongs to. The third type of the fuzzy classifier has the following form of consequents:

$$R_k\colon\ \ldots, \text{THEN class is } y_{o(k(i))} \text{ with } CF_{k,o(k(i))}, \tag{3}$$

where $y_{o(k(i))}$ indicates the class that the $o(k(i))$-th input data belongs to, $k(i)$ denotes the all possible values with the scope of the output variable, that is, $y_{o(k(i))}$ represents all possible classes, and $CF_{k,o(k(i))} \in [0, 1]$ refers to the degree of confidence.

In this paper, we adopt the form in (3) for the consequents of fuzzy rules. As such, the fuzzy classification system can be formulated in detail as follows. Suppose that there are n samples belonging to M classes. For the fuzzy classifier, there are m training samples with $x_p = (x_{p1}, x_{p2}, \ldots, x_{pm})$ being the p-th $(1 \le p \le m)$ training data, and $y_p \in \{1, 2, \ldots, M\}$ denotes the class that x_p belongs to. The problem of classifying the n samples into M classes can be defined as follows. For the input variable $\overline{x} = x_1, x_2, \ldots, x_m$, the IF-THEN rules for the fuzzy classifier are given by

$$\begin{aligned} R_{(j_1,j_2,\ldots,j_n)}\colon\ &\text{IF } x_1 \text{ is } A_{(1,j_1)} \text{ and } x_2 \text{ is } A_{(2,j_2)} \text{ and, } \ldots, x_n \text{ is } A_{(n,j_n)}, \\ &\text{THEN } \overline{x} \text{ belong to Class } y_{(j_1,j_2,\ldots,j_n)} \text{ with } CF = CF_{(j_1,j_2,j_n)}, \end{aligned} \tag{4}$$

where $A_{(1,j1)}$ denotes the j_i-th fuzzy set for the input variable (or feature variable) x_i in the antecedent of the fuzzy rule, $y_{(j1,j2,\ldots,jn)} \in \{1, 2, \ldots M\}$ denotes the output variable of the consequent of the fuzzy rule (i.e., the class the sample belongs to), and $CF_{(j1,j2,\ldots,jn)}$ represents the degree of confidence for the IF-THEN fuzzy rule, $R_{(j1,j2,\ldots,jn)}$.

For the fuzzy classification system represented by (4), we define the membership function of the input variable x_i in the antecedent of the IF-THEN rule on fuzzy set j_i:

$$\mathrm{A}_{(i,ji)}\left(c_{(i,ji)}, w^l_{(i,ji)}, w^r_{(i,ji)}; x_i\right) = \begin{cases} \exp\left[-\left(\frac{x_i - c_{(i,ji)}}{w^l_{(i,ji)}}\right)^2\right]; & \text{if } x_i \le c_{(i,ji)}, \\ \exp\left[-\left(\frac{x_i - c_{(i,ji)}}{w^r_{(i,ji)}}\right)^2\right]; & \text{if } x_i \le c_{(i,ji)}. \end{cases} \tag{5}$$

This membership is essentially Gaussian one including three parameters, $c_{(i,ji)}$ denoting the center of the membership function and $w^l_{(i,ji)}$ and $w^r_{(i,ji)}$ representing the left

and right widths of the membership function, respectively, which are not necessarily equal. We can select the value of the three parameters to determine the membership function for the antecedent of the fuzzy rules.

The center of the membership function $c_{(i,ji)}$ is the center of the fuzzy partition. When selecting $w^l_{(i,ji)}$ and $w^r_{(i,ji)}$, we introduce a constantμ, which is known as an overlap factor and reflects the overlap degree of the membership:

$$\begin{cases} A_{(i,ji)}\left(c_{(i,ji-1)}\right) = \exp\left[-\left(\frac{x_i - c_{(i,ji)}}{w^l_{(i,ji)}}\right)^2\right] = \mu, \\ A_{(i,ji)}\left(c_{(i,ji+1)}\right) = \exp\left[-\left(\frac{x_i - c_{(i,ji)}}{w^r_{(i,ji)}}\right)^2\right] = \mu, \end{cases} \tag{6}$$

in order to improve the interpretability of membership functions. Thus $w^l_{(i,ji)}$ and $w^r_{(i,ji)}$ can be determined by

$$\begin{cases} w^l_{(i,ji)} = \left[-\frac{\left(c_{(i,ji-1)} - c_{(i,ji)}\right)^2}{\ln(\mu)}\right]^{(1/2)}, \\ w^r_{(i,ji)} = \left[-\frac{\left(c_{(i,ji+1)} - c_{(i,ji)}\right)^2}{\ln(\mu)}\right]^{(1/2)}. \end{cases} \tag{7}$$

3. The Proposed QIQPSO

In this paper, we propose a quantum-inspired quantum-behaved particle swarm optimization (QiQPSO), of which the position of particle i at the t-th iteration is represented as a string of qubits. Unlike the bits representing the information in either 0's or 1's in the classical computing, in quantum computing, a qubit could take the value of 0, 1, or a superposition of both [33]. Superposition allows the possible states to represent both 0 and 1 simultaneously based on its probability. In this section, we first give a brief introduction of QPSO and then propose the QiQPSO algorithm.

3.1. A Brief Introduction to QPSO. Quantum-behaved particle swarm optimization (QPSO) is a variant of PSO motivated by quantum mechanics and the trajectory analysis of the canonical PSO [22, 23, 34, 35, 36]. In the QPSO with L individuals, each individual is treated as a volume-less and spin-less particle in the N-dimensional space, with the current position vector of particle i at the t-th iteration represented by $X_{i,t} = (X^1_{i,t}, X^2_{i,t}, \ldots, X^N_{i,t})$. Besides, each particle i also has a vector $P_{i,t} = (P^1_{i,t}, P^2_{i,t}, \ldots, P^N_{i,t})$, called personal best (pbest) position, which is the best previous position (the position giving the best objective function value or fitness value) of particle i. The whole swarm has vector $G_t = (G^1_t, G^2_t, \ldots, G^N_t)$, known as the global best (gbest) position, which is the position of the best particle among all the particles in the population. Consider the following minimization problem:

$$\text{minimize } f(X), \text{s.t} \quad X \in S \subseteq R^N, \tag{8}$$

where $f(X)$ is an objective function continuous almost everywhere and S is the feasible space. Note that $f(X)$ always has many local minima, which leads to its difficulty in finding a global minima when minimizing it. Then, $P_{i,t}$ can be updated by

$$P_{i,t} = \begin{cases} X_{i,t} & \text{if } f\left(X_{i,t}\right) < f\left(P_{i,t-1}\right), \\ P_{i,t-1} & \text{if } \left(X_{i,t}\right) \geq f\left(P_{i,t-1}\right), \end{cases} \tag{9}$$

and accordingly G_t can be found by $G_t = P_{g,t}$, where $g = \underset{1 \leq i \leq L}{\operatorname{argmin}}[f(P_{i,t})]$. The particle in the QPSO updates its position in the following equation [22-27]:

$$X^j_{i,t+1} = p^j_{i,t} \pm \gamma \left| X^j_{i,t} - C^j_t \right| \ln\left(\frac{1}{u^j_{i,t+1}}\right), \tag{10}$$

for $1 \leq i \leq L$ and $1 \leq j \leq N$, where

$$\begin{aligned} p^j_{i,t} &= \varphi^j_{i,t} P^j_{i,t} + \left(1 - \varphi^j_{i,t}\right) G^j_t, \\ \varphi^j_{i,t} &\sim U(0,1). \end{aligned} \tag{11}$$

In (10), $C_t = (C^1_t, C^2_t, \ldots, C^N_t)$ is called the mean best (mbest) position defined by the average of the pbest positions of all particles, i.e., $C^j_t = (1/L)\sum_{i=1}^{L} P^j_{i,t}$, $(1 \leq j \leq N)$, $u^j_{i,t+1}$ Is a sequence of random numbers uniformly distributed on (0, 1), and parameter γ is known as the contraction-expansion (CE) coefficient, which can be adjusted to balance the local and global search of the algorithm during the optimization process.

3.2. Quantum-Inspired QPSO. In the quantum computing and quantum information, the quantum state is modeled by the Hilbert space of wave functions and is defined as $|\alpha|^2$

$$|\psi\rangle = \alpha|1\rangle + \beta|0\rangle, \tag{12}$$

where α and β are the complex numbers defining probabilities at which the corresponding state is likely to appear when a qubit collapses, for instance, when reading or measuring. Probability fundamentals stated that

$$|\alpha|^2 + |\beta|^2 = 1, \tag{13}$$

where $|\alpha|^2$ gives the probability that a qubit is in the OFF (0) state and $|\beta|^2$ gives the probability that a qubit is in the ON (1) state. The probability of $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ can be represented as a quantum angle θ, where $\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ satisfies the probability fundamental of

$$|\sin(\theta)|^2 + |\cos(\theta)|^2 = 1, \tag{14}$$

where the parameter θ has been used to calculate and update the probability in quantum-inspired EA [29, 30] and quantum-inspired PSO [33].

In this paper, we proposed a quantum-inspired quantum-behaved particle swarm optimization. In this method, the position of particle i at the t-th iteration is represented as

a string of qubits, which corresponds to a vector of quantum angles $\theta_{i,t} = (\theta_{i,t}^1, \theta_{i,t}^2 \ldots, \theta_{i,t}^N)$, where $\theta_{i,t}^j$ denotes the quantum angle of the j-th qubit $(1 \le j \le N)$ and the vector of the quantum angles corresponding to the personal best position of particle i is $\theta p\text{best}_{i,n} = (\theta p\text{best}_{i,t}^1, \theta p\text{best}_{i,t}^2 \ldots, \theta p\text{best}_{i,t}^N)$.

The vector of quantum angles corresponds to the global best position of the swarm and is denoted as $\theta g\text{best}_t = (\theta g\text{best}_t^1, \theta g\text{best}_{i,t}^2, \ldots, \theta g\text{best}_{i,t}^N)$. Thus, according to equation (10), the quantum angle for each qubit is updated as

$$\theta_{i,t+1}^j = \theta p_{i,t}^j \pm \gamma \left| \theta m\text{best}_t^j - \theta_{i,t}^j \right| \ln\left(\frac{1}{\mu_{i,t+1}^j}\right), \tag{15}$$

where $\theta m\text{best}_t^j$ is the quantum angle corresponding to the j-th qubit of the mbest position of the swarm at the t-th iteration. However, we have to obtain the increment of the quantum angle to update the probability by using a rotation gate. Thus, we further get the increment of the quantum angle for each qubit by

$$\Delta\theta_{i,t+1}^j = \theta_{i,t+1}^j - \theta_{i,t}^j = \theta p_{i,t}^j \pm \gamma \left| \theta m\text{best}_t^j - \theta_{i,t}^j \right| \ln\left(\frac{1}{\mu_{i,t+1}^j}\right) - \theta_{i,t}^j, \tag{16}$$

with which we can update the probability corresponding to each qubit by the following qubit rotation gate:

$$\begin{bmatrix} \alpha_{i,t+1}^j \\ \beta_{i,t+1}^j \end{bmatrix} = \begin{bmatrix} \cos\left(\Delta\theta_{i,t+1}^j\right) & -\sin\left(\Delta\theta_{i,t+1}^j\right) \\ \sin\left(\Delta\theta_{i,t+1}^j\right) & \cos\left(\Delta\theta_{i,t+1}^j\right) \end{bmatrix} \begin{bmatrix} \alpha_{i,t}^j \\ \beta_{i,t}^j \end{bmatrix}, \tag{17}$$

where $\begin{bmatrix} \alpha_{i,t}^j \\ \beta_{i,t}^j \end{bmatrix}$ represents the probability of the j-th qubit of particle i at the t-th iteration, with which we can collapse the corresponding qubit to be "0" or "1".

4. Optimizing Fuzzy Classifier with QiQPSO

When using QiQPSO to optimize a fuzzy rule-based classifier, we first construct the initial fuzzy classifier with QiQPSO based on the fuzzy rule base obtained directly from the samples by using the grid partition method, and then we further optimize the fuzzy rule base to reduce the number of the rules with the classification accuracy unchanged. In the section, we describe these two procedures in detail.

4.1. Construction of the Initial Fuzzy Classification System with QiQPSO. When using QiQPSO to construct a fuzzy rule-based classifier, we should determine the antecedents of the fuzzy IF-THEN rules firstly. Then, the determined antecedent trains the known data to achieve the consequent of the fuzzy rules. The antecedent of the fuzzy rules contains information about the number of input variables, the partition of input space, and the number of input space partitions. Here, we present how to use QiQPSO to establish the initial fuzzy rule-based classifier, by adopting Gaussian membership functions and using the grid method for partitioning the input space (i.e., the feature space).

4.1.1. Determining Antecedent of Fuzzy Rules

(1) *Selection of Feature Variables.*

For the purpose of improving the interpretability of a fuzzy system model, the number of the selected input variables (i.e., feature variables) should be as few as possible. The method employed in this paper is to implement feature variable selection by using QiQPSO during the construction of the initial fuzzy classification system. To this end, we represent each particle in QiQPSO as a string of qubits, with the first n (n is the number of the feature variables) qubits, each reflecting whether the corresponding feature variable is selected or not. If the j-th $(1 \le \mathbf{j} \le \mathbf{n})$ bit is "1", then the j-th feature variable (or input variable) is selected; otherwise, the feature variable is not selected.

(2) *Fuzzy Partition of the Feature Space.*

We adopt the following procedure, i.e., the grid method, to partition the feature space. First, the feature space is divided evenly, with the number of partitions and the center of each partition determined. Then, QiQPSO is employed to select the midpoints to implement fuzzy partition of the feature variables. Specifically, suppose that the domain of the j-th feature variable $\mathbf{x_j}$ has $\mathbf{K_j}$ even partitions, $\left\{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_{\mathbf{K_j}}\right\}$, which is a qubit string composed of 0's and 1's. If a bit is "1," the center of this even partition is selected as the valid center of the fuzzy partition; otherwise, the center is not selected. The number of all the valid centers in this string, namely, the sum of all 1's in the string, is known as the number of fuzzy partitions $\mathbf{k_j}$.

Let $\mathbf{x_j^{min}}$ and $\mathbf{x_j^{max}}$ be the maximum and the minimum of the feature variable $\mathbf{x_j}$. The k-th center of the fuzzy partition of $\mathbf{x_j}$ is then given by

$$\mathrm{c}(\mathrm{j},\mathrm{k}) = \mathrm{x_j^{min}} + \left(\mathrm{I_{jk}}\text{-}1\right)\frac{\mathrm{x_j^{max}}\text{-}\mathrm{x_j^{min}}}{\mathrm{k_j}\text{-}1}, \quad \mathrm{k} \in \left\{1, 2, \ldots, \mathrm{k_j}\right\}, \tag{18}$$

where $\mathbf{I_{jk}}$ is the index of the k-th bit whose value is "1" in the string. The center of the fuzzy partition is regarded as the center of the membership function. In order to guarantee the interpretability of the membership function, according to (6), we have

$$A_{(j,k)}\left(c_{(j,k-1)}\right) = \exp\left\{-\left[\frac{c_{(j,k-1)} - c_{(j,k)}}{w_{(j,k)}^l}\right]^2\right\} = \mu, \tag{19}$$

$$A_{(j,k)}\left(c_{(j,k+1)}\right) = \exp\left\{-\left[\frac{c_{(j,k+1)} - c_{(j,k)}}{w_{(j,k)}^r}\right]^2\right\} = \mu, \tag{20}$$

which means the membership function overlaps at constant μ as shown in Figure 1.

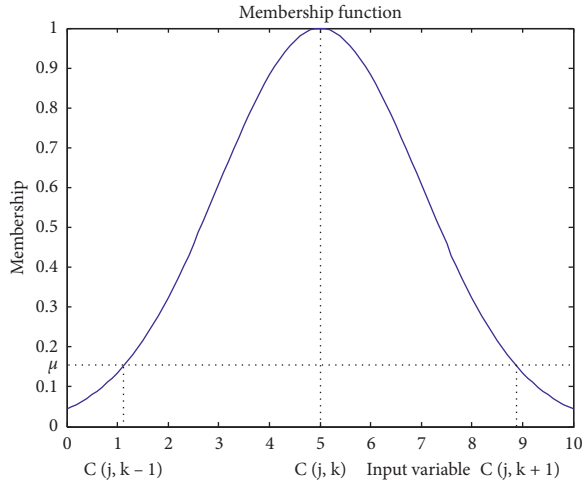

Figure 1: Overlap degree of the membership function.

Thus, according to (7), the left width $w^l_{j,k}$ and the right width $w^r_{j,k}$are given by

$$w^l_{(j,k)} = \sqrt{\frac{\left(c_{(j,k)} - c_{(j,k-1)}\right)^2}{-\ln(\mu)}}, \tag{21}$$

$$w^r_{(j,k)} = \sqrt{\frac{\left(c_{(j,k+1)} - c_{(j,k)}\right)^2}{-\ln(\mu)}}, \tag{22}$$

where μ is the overlap factor and it guarantees that when the k-th membership function value is μ, its corresponding variable value coincides with the centers of the (k-1)th and the (k+1)th membership functions. Such membership functions generally have high interpretability.

4.1.2. Encoding of the Particle Position in QPSO. The position of each particle is encoded as a qubit string with two sections, one of which represents the feature variable selection and the other of which reflects the selection of the centers of feature space partitions. As mentioned above, each qubit in a particle position string has a corresponding quantum angle which is updated and then determines the probability of the qubit value according to equations (16) and (17), respectively.

4.1.3. Encoding of the Particle Position in QiQPSO. An important objective in constructing the initial fuzzy classification system is to maximize the number of the correctly classified samples. Thus, the classification accuracy should be one of the most important components in the fitness function. Another objective is to minimize the number of the selected features as the number of the fuzzy rules increases exponentially with the feature number. Thus, the number of the selected feature variables should be another component in the fitness function. The other component in the fitness function is the number of the fuzzy rules since too many rules reduce the interpretability of the fuzzy classifier.

As such, the fitness function for the construction of the initial fuzzy classification system with the QiQPSO is given by

$$N_r = \prod_{j=1}^{N_f} k_j. \tag{23}$$

4.1.4. Determining the Consequents of Fuzzy Rules. As mentioned in Section 2, the consequents of the fuzzy rules include the index y of the class that the sample belongs to and the corresponding degree of confidence. When using the QiQPSO algorithm to construct the initial fuzzy classifier, we can determine the antecedents of the fuzzy rules for a given collapsed qubit string of the particle and then obtain the consequents of the fuzzy rules by the following procedure.

First, according to the antecedents of fuzzy rules and the sample data, we determine $\boldsymbol{\beta}_{\mathbf{CT}}$ that reflects the extent to which the samples belong to class T:

$$\beta_{CT} = \sum_{X_p \in CT} \prod_{i=1}^{n} A_{(i,ji)}(x_i), \quad T \in \{1, 2, \ldots, M\}. \tag{24}$$

Then, we can obtain the class index y in the consequent of the fuzzy rules by using the following formula:

$$Y = \overset{\text{argmax}}{1 \le T \le M} \beta_{CT}, \tag{25}$$

$$y_{(j_1, j_2, \ldots, j_n)} = Y. \tag{26}$$

Finally, we can compute the corresponding degree of confidence as

$$CF_{(j1,j2,\ldots,jn)} = \frac{\beta_{CY} - \beta}{\sum_{T=1}^{M} \beta_{CT}}, \tag{27}$$

where

$$\beta = \frac{\sum_{T=1}^{M} \beta_{CT}}{(M-1)}. \tag{28}$$

4.1.5. Procedure of Constructing the Initial Fuzzy Classifier with QiQPSO. The procedure of constructing the initial fuzzy classifier with QiQPSO is described as follows:

Step 1: set the parameters of the QiQPSO algorithm: the number of maximum iterations of the algorithm $T_{\max}$, the swarm size L, the overlapping factor, and the weights of the components in the fitness function, etc.

Step 2: set the number of the iteration $t = 1$ and randomly initialize the particle swarm, namely, randomly initialize the quantum angle of each qubit of each particle and collapse each qubit to be "0" or "1".

Step 3: do the following procedures.

Step 4: while $i \le L$, where i is the particle index, do the following procedures.

Step 5: according to the qubit string of the k-th particle, determine the selected feature variables, the fuzzy partition of the feature space, and thus the antecedents of the fuzzy rules.

Step 6: calculate the parameters of the consequents of the fuzzy rules according to equations (24) to (28).

Step 7: calculate the fitness function values of the string of each particle according to (22).

Step 8: update $\theta p\text{best}_{i,t}$ and $\theta g\text{best}_t$ accordingly.

Step 9: calculate the increment of each quantum angle according to (16), update the probability of each qubit according to (17), and collapse each qubit to "0" or "1" accordingly.

Step 10: set $i = i + 1$. If $i \le L$, return to Step 4; otherwise, continue to the next step.

Step 11: set $t = t + 1$. If $t \le T_{max}$, go to Step 3; otherwise, continue to the next step.

Step 12: output the collapsed qubit string of the particle with the highest fitness value and determine the fuzzy classification model based on this binary string.

4.2. Optimizing the Fuzzy Classifier with QiQPSO. The initial fuzzy classification system constructed by QiQPSO may have many redundant and invalid fuzzy rules since the feature space for the membership functions is partitioned by the grid method. Thus, we need to optimize the fuzzy classifier by removing the redundant and invalid rules without changing the interpretability of the classifier. In this paper, QiQPSO is also employed to achieve this purpose.

4.2.1. Encoding of the Particle Position. Each particle position is encoded as a qubit string with each qubit having a corresponding quantum angle. The value of each collapsed qubit represents whether a fuzzy rule should be included in the fuzzy classifier or not. If the qubit is "1," the corresponding fuzzy rule is selected; otherwise, it is removed from the fuzzy rule base of the classifier.

4.2.2. Fitness Function. The purpose of optimizing the fuzzy rule base of the fuzzy classifier is to delete the redundant and invalid rules in order to minimize the number of the fuzzy rules without reducing the classification accuracy of the classifier. Thus, there should be two components in the fitness function, one of which is the number of the correctly classified samples and the other of which is the number of fuzzy rules. As a result, the following fitness function is employed for fuzzy rule base optimization:

$$F_2(I) = \omega_4 N_c - \omega_5 N_r, \tag{29}$$

where ω_4 and ω_5 are the weights of the two components.

4.2.3. The Procedure of the Fuzzy Rule Base Optimization with QiQPSO. With the above specification, the procedure of optimizing the fuzzy rule base of the classifier is presented as follows:

Step 1: set the parameters of the QiQPSO algorithm: the number of maximum iterations of the algorithm $T_{\max}$, the swarm size L, the weights of the components in the fitness function, etc.

Step 2: set the number of the iteration $t = 1$ and randomly initialize the particle swarm, namely, randomly initialize the quantum angle of each qubit of each particle and collapse each qubit to be "0" or "1".

Step 3: do the following procedure.

Step 4: for each particle i, do the following procedure.

Step 5: according to the collapsed qubit string of the i-th particle, determine the fuzzy rules selected into the fuzzy classifier and find the number of selected fuzzy rules N_r.

Step 6: classify all the samples by the given fuzzy classifier and determine the number of the correctly classified samples N_c.

Step 7: calculate the fitness function values of the collapsed qubit string of each particle according to (29).

Step 8: update $\theta p\text{best}_{i,t}$ and $\theta g\text{best}_t$ accordingly.

Step 9: calculate the increment of each quantum angle according to (16), update the probability of each qubit according to (17), and collapse each qubit to "0" or "1" accordingly.

Step 10: set $i = i + 1$. If $i \le L$, return to Step 4; otherwise, continue to the next step.

Step 11: set $t = t + 1$. If $t \le T_{\max}$, go to Step 3; otherwise, continue to the next step.

Step 12: output the binary string of the particle with the highest fitness value and determine the fuzzy classification model based on this binary string.

5. Experimental Results

In this section, we first introduce the data sets used and parameter setting in experiments in detail in Section 5.1. In Section 5.2, experimental results for each data set are discussed separately, and we also had done experiments on those data sets with other state-of-the-art algorithms and compared them with the proposed algorithm QiQPSO.

5.1. Data Sets and Experiment Settings. The experiments were carried out on various high-dimensional classification problems to establish and optimize fuzzy classifiers for these data sets by using the proposed QiQPSO algorithm, and Table 1 summarizes the most important characteristics of these data sets. Iris data consist of 150 samples with four dimensions (namely, calyx length, calyx width, petal length, and petal width) and is divided into three classes, each of which is composed of 50 samples.

Table 1: Summary of experiment data sets.

Data set name	Patterns	Features	Classes
Iris	150	4	3
Wine	178	13	3
Glass	210	9	6
Seeds	210	7	3
Wdbc	569	30	2
Satimage	6435	36	6

The samples of the first and the second classes are separated completely, while the second and the third classes intersect. The Wine data set has 178 samples of chemical compositions of three different wines, each sample having 13 attributes. The Glass data set is composed of 9-dimensional 210 samples that can be classified into 6 categories. The Seeds data set consists of 210 data that can be divided into three categories and each sample has seven attributes. In order to verify the performance of the proposed algorithm on more complex real-world problems, Wdbc and Satimage are selected as testing data sets. These two data sets are both high-dimension real-world problems from the UCI repository [11] containing 569 and 6435 samples, respectively.

In all the experiments, the maximum number of iterations that the QiQPSO algorithm executed for was set to be 100 and the swarm size was $L = 20$. Note that these two parameters are set empirically according to the dimension of the optimization problems. That is, higher-dimensional problems generally needs larger maximum number of iterations and larger swarm size. The CE coefficient of the algorithm decreased linearly from 1.0 to 0.5. When constructing the initial fuzzy classifiers, the weights in the fitness function was set to be $\omega_1 = 1$, $\omega_2 = 0.5$, and $\omega_3 = 2$. The overlap factor for the membership functions was set as $\mu = 0.2$. When the fuzzy rule bases were optimized with QiQPSO, the weights in the fitness function were set as $\omega_4 = 1$ and $\omega_5 = 2$.

5.2. Experimental Results and Analysis

5.2.1. Results for Iris Data Set. For the Iris data set, we denote the four features, i.e., calyx length, calyx width, petal length, and petal width, as x_1, x_2, x_3, and x_4, respectively. The initial fuzzy classifier constructed by QiQPSO for this data is shown in Table 2. It can be seen that the initial classification system only uses two feature variables, namely, x_3 and x_4, both of which employ four Gaussian membership functions to divide the feature space as shown in Figure 2. Thus, the initial classifier has 16 fuzzy rules in total.

From Table 2, it is shown that the initial fuzzy classifier for the Iris data set contains two invalid fuzzy rules (i.e., R13 and R14), suggesting that there is no data under these two fuzzy rules. Using this fuzzy classifier, we can find that the number of the correctly classifier samples is 143 although there are intersections between the second and the third classes.

The final fuzzy classifier further optimized by QiQPSO is presented in Table 3, in which only 9 fuzzy rules are included. More specifically, the rules in the final fuzzy classifier are

$$\begin{aligned} &R1\text{: IF } x3 \text{ is } A(4,1) \text{ and } x4 \text{ is } B(4,1)\\ &\text{THEN } x \text{ belongs to Class 1 with } CF = 1.0\\ &R2\text{: IF } x3 \text{ is } A(4,1) \text{ and } x4 \text{ is } B(4,2)\\ &\qquad \cdots \quad \cdots\\ &R15\text{: IF } x3 \text{ is } A(4,4) \text{ and } x4 \text{ is } B(4,3)\\ &\text{THEN } x \text{ belongs to Class 3 with } CF = 0.7510\\ &R16\text{: IF } x3 \text{ is } A(4,4) \text{ and } x4 \text{ is } B(4,4)\\ &\text{THEN } x \text{ belongs to Class 3 with } CF = 1.0. \end{aligned} \tag{30}$$

The final fuzzy classifier further optimized by QiQPSO was then tested on the Iris data set. Although the classification accuracy remains the same, the number of the fuzzy rules reduced from 16 to 9 after the fuzzy rule base of the initial classifier was optimized. This means that after the rule base was optimized by QiQPSO, the redundant and invalid fuzzy rules were removed so that the interpretability of the classifier is improved.

5.2.2. Results for Wine Data Set. For the Wine data set, the 13 attributes (i.e., compositions) are alcohol (x_1), malic acid (x_2), ash content (x_3), ash content alkalinity (x_4), magnesium (x_5), total phenol (x_6), falconoid (x_7), phenolic (x_8), anthocyanin (x_9), color intensity (x_{10}), chroma (x_{11}), OD280/OD315 dilute wine (x_{12}), and amino acid (x_{13}). Among them, alcohol (x_1), flavonoid (x_7), chroma (x_{11}), OD280/OD315 dilute wine (x_{12}), and amino acid (x_{13}) are the input variables of the initial fuzzy classification system established by QiQPSO, and the feature space of each variable is partitioned by using Gaussian membership functions as shown in Figure 3.

For the wine data set, the initial fuzzy classifier consists of 32 fuzzy rules with five input variables. It can correctly classify 161 samples. After the fuzzy rule base was optimized by QiQPSO, the improved classifier contains only 10 fuzzy rules. More specifically, these rules are expressed as

Table 2: The initial fuzzy classification system constructed by QiQPSO for Iris.

RU	Input x_3			Input x_4			C	CF
	Cen	L	R	Cen	L	R		
1	1	0	0.9654	0.1	0	0.6548	1	1.0
2	1	0	0.9654	0.7	0.6548	0.6548	1	1.0
3	1	0	0.9654	1.3	0.6548	0.6548	2	0.6842
4	1	0	0.9654	2.4	0.6548	0	1	0.9385
5	2.5	0.9654	0.9654	0.1	0	0.6548	1	1.0
6	2.5	0.9654	0.9654	0.7	0.6548	0.6548	1	1.0
7	2.5	0.9654	0.9654	1.3	0.6548	0.6548	2	0.3335
8	2.5	0.9654	0.9654	2.4	0.6548	0	2	0.3424
9	4.0	0.9654	0.9654	0.1	0	0.6548	2	0.7506
10	4.0	0.9654	0.9654	0.7	0.6548	0.6548	2	1.0
11	4.0	0.9654	0.9654	1.3	0.6548	0.6548	2	0.8935
12	4.0	0.9654	0.9654	2.4	0.6548	0	3	1.0
13	5.9	0.9654	0	0.1	0	0.6548	0	NULL
14	5.9	0.9654	0	0.7	0.6548	0.6548	0	NULL
15	5.9	0.9654	0	1.3	0.6548	0.6548	3	0.7510
16	5.9	0.9654	0	2.4	0.6548	0	3	1.0

RU = rules, Cen = center, L = left width, R = right width, C = classification, and CF = confidence.

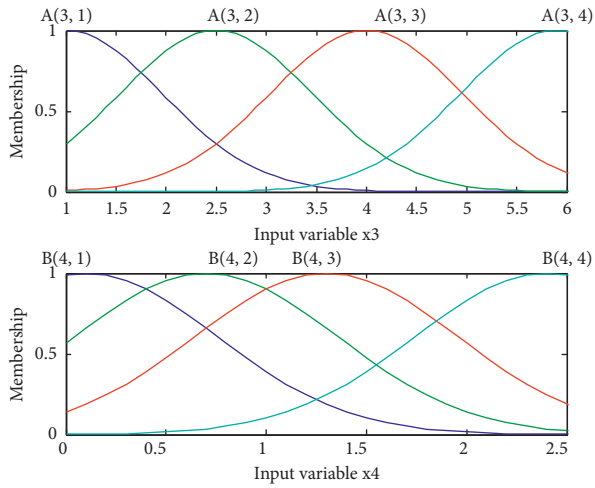

Figure 2: Membership function of the Iris initial fuzzy classification system.

Table 3: The final fuzzy classifier with the fuzzy rule base further optimized by QiQPSO for the Iris data set.

RU	Input x_3			Input x_4			C	CF
	Cen	L	R	Cen	L	R		
1	1	0	0.9654	0.1	0	0.6548	1	1.0
2	1	0	0.9654	0.7	0.6548	0.6548	1	1.0
5	2.5	0.9654	0.9654	0.1	0	0.6548	1	1.0
6	2.5	0.9654	0.9654	0.7	0.6548	0.6548	1	1.0
10	4.0	0.9654	0.9654	0.7	0.6548	0.6548	2	1.0
11	4.0	0.9654	0.9654	1.3	0.6548	0.6548	2	0.8935
12	4.0	0.9654	0.9654	2.4	0.6548	0	3	1.0
15	5.9	0.9654	0	1.3	0.6548	0.6548	3	0.7510
16	5.9	0.9654	0	2.4	0.6548	0	3	1.0

RU = rules, Cen = center, L = left width, R = right width, C = classification, and CF = confidence.

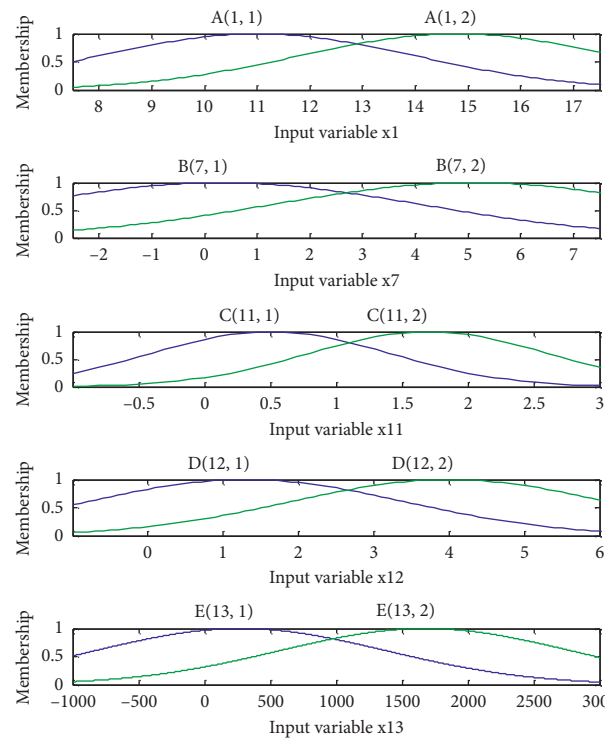

Figure 3: Membership functions of the initial fuzzy classification system for the Wine data set.

$$
\begin{aligned}
R1&: \text{IF } x1 \text{ is } A(1,1) \text{ and } x7 \text{ is } A(7,1) \text{ and } x11 \text{ is } A(11,1)\\
&\text{and } x12 \text{ is } A(12,1) \text{ and } x13 \text{ is } A(13,1)\\
&\text{THEN } x \text{ belongs to Class 3 with } CF = 1.0\\
R2&: \text{IF } x1 \text{ is } A(1,1) \text{ and } x7 \text{ is } A(7,1) \text{ and } x11 \text{ is } A(11,1)\\
&\text{and } x12 \text{ is } A(12,1) \text{ and } x13 \text{ is } A(13,1)\\
&\text{THEN } x \text{ belongs to Class 3 with } CF = 1.0\\
&\cdots \quad \cdots\\
R15&: \text{IF } x1 \text{ is } A(1,1) \text{ and } x7 \text{ is } A(7,1) \text{ and } x11 \text{ is } A(11,2)\\
&\text{and } x12 \text{ is } A(12,2) \text{ and } x13 \text{ is } A(13,1)\\
&\text{THEN } x \text{ belongs to Class 1 with } CF = 0.9900\\
R16&: \text{IF } x1 \text{ is } A(1,2) \text{ and } x7 \text{ is } A(7,1) \text{ and } x11 \text{ is } A(11,1)\\
&\text{and } x12 \text{ is } A(12,1) \text{ and } x13 \text{ is } A(13,1)\\
&\text{THEN } x \text{ belongs to Class 1 with } CF = 0.9977.
\end{aligned}
\tag{31}
$$

The comparison is between the initial fuzzy classification system and the one improved by QiQPSO. It is shown that the initial classification system established by QiQPSO has 5 input variables, 10 fuzzy sets for the antecedents, and 32 fuzzy rules. With this initial classifier, the classification accuracy is 90.45% for the data set. After the fuzzy rule base

was optimized by QiQPSO, the number of fuzzy rules reduced to 10 with the classification accuracy unchanged.

5.2.3. Results for Glass Data Set. Each sample in the Glass data set has nine features. However, after optimized by QiQPSO, the initial fuzzy classifier only needs five features, i.e., x_3, x_4, x_6, x_8, and x_9, and the feature space of each variable is partitioned by 2 Gaussian membership functions. Thus, the initial fuzzy classifier established by the algorithm has $2^5 = 32$ fuzzy rules. Due to the space limitation, these fuzzy rules are not listed here. After the fuzzy rule base was optimized by QiQPSO, the improved classifier only needs 17 fuzzy rules. Comparing to the initial fuzzy classification system, the one with the fuzzy rule base optimized by QiQPSO can classify the sample data with an accuracy of 84.11% the same as the initial fuzzy classification system, but the optimized classifier only has 17 fuzzy rules, which means that the interpretability of the classifier is enhanced by QiQPSO.

5.2.4. Results for Seeds Data Set. Each sample in the Seeds data set has 7 features. After feature selection by the QiQPSO algorithm, the dimensionality of the data is reduced to 5, only x_1, x_2, x_4, x_5, and x_7 being selected as the input variables of the initial fuzzy classification system. The feature space of each input variable is partitioned by 3 Gaussian membership functions so that the initial fuzzy classifier has $3^5 = 243$ fuzzy rules, which are not given here due to the space limitation. The fuzzy rule base was further optimized by QiQPSO, and the resulting fuzzy classifier only needs 12 rules.

The classifiers classify the sample data with an accuracy of 84.86%, but the one with the fuzzy rule base optimized only has 12 rules, compared to the 243 rules in the initial system.

5.2.5. Results for Wdbc Data Set. Each sample in the Wdbc data set has 64 features. However, after being optimized by QiQPSO, the initial fuzzy classifier only needs 10 features, and the feature space of each variable is partitioned by 3 Gaussian membership functions. Thus, the initial fuzzy classifier established by the algorithm has $3^{10} = 59049$ fuzzy rules. After the fuzzy rule base was optimized by QiQPSO, the improved classifier only needs 97 fuzzy rules. It is shown that both classifiers can classify the sample data with an accuracy of 79.31%, but the optimized classifier only has 200 fuzzy rules, which means that the interpretability of the classifier is enhanced after it is the QiQPSO.

5.2.6. Results for Satimage Data Set. Each sample in the Satimage data set has 36 features. However, after optimized by QiQPSO, the initial fuzzy classifier only needs 10 features, and the feature space of each variable is partitioned by 3 Gaussian membership functions. Thus, the initial fuzzy classifier established by the algorithm has 3^{10} fuzzy rules. After the fuzzy rule base was optimized by QiQPSO, only 153 fuzzy rules were selected for the final classification. The accuracy of these two algorithms is both 81.53%, but the optimized classifier only has 153 fuzzy rules, implying that the optimized classifier has higher interpretability than the original one.

5.2.7. Performance Comparison with Other Methods. In this section, we first compared the proposed algorithm to C4.5, and then we used other optimization methods, namely, genetic algorithm (GA) and the binary PSO (BPSO) [37], to establish the initial fuzzy classifiers and further optimize the fuzzy rule bases for the six data sets and to make a performance comparison between these methods and QiQPSO. Just as the construction of the initial fuzzy classifiers with QiQPSO, we also used GA or BPSO to build the initial fuzzy classification systems based on the system directly established from the sample data.

We all know that using particle swarm optimization to optimize a problem enhances the computational complexity. However, as shown in Table 4, although the time complexity of the proposed algorithm is not better than C4.5, the proposed algorithm has a much higher classification accuracy than C4.5. Therefore, it is still acceptable although the time complexity is higher.

The parameter settings for GA and BPSO are as follows. For GA, the crossover probability was set to be 0.5 and the mutation probability was 0.05. The population size was 50, and the maximum number of generations was also 100. For BPSO, the swarm size and the maximum number of iterations were set to be 20 and 100, respectively. The fitness functions for constructing the initial classifiers and optimizing the fuzzy rule bases were the same as those used in the experiments for QiQPSO. The experimental results are shown in Table 5, where the results of the fuzzy classifier established directly from the sample data and further optimized by GA are also listed for comparison.

For the Iris data set, the fuzzy classifier constructed directly from the sample data has 256 rules. After optimized by GA, the number of the rules decreases to 26 with a classification accuracy of 95.33%. The fuzzy classification system constructed by GA consists of 16 rules, and the final classifier improved by the algorithm only needs 12 rules that correctly classify 94.67% of the samples. The classifier constructed and further optimized by BPSO has the same number of the final fuzzy rules and the same classification accuracy as those by GA. QiQPSO generated the best classifier with the least number of rules and the highest classification accuracy (95.33%).

For the Wine data set, the fuzzy classifier constructed directly from the sample data and optimized by GA has the highest classification accuracy (95.51%) but needs 70 fuzzy rules. QiQPSO obtained the final classifier with the second highest classification accuracy,, but only 10 fuzzy rules are needed to achieve this classification accuracy. GA and BPSO also showed the same performance in constructing and optimizing the fuzzy classifier for the data set, but their performance is obviously poorer than that of QiQPSO.

For the Glass data set, the fuzzy classifier constructed directly from the samples and optimized by GA has the highest classification accuracy (87.85%), but it needs 118

Table 4: Comparison between the initial fuzzy classifier and the optimized one for the Iris data set.

Data set name	Fuzzy classification system (FCS)	The number of feature variables	The number of fuzzy sets	The number of fuzzy rules	The number of correctly classified samples	Classification accuracy (%)
Iris	Initial FCS	2	8	16	143	95.33
	Optimized FCS	2	8	9	143	95.33
	C4.5	2	8	9	138	92.00
Wine	Initial FCS	5	10	32	161	90.45
	Optimized FCS	5	10	10	161	90.45
	C4.5	5	10	10	159	89.32
Glass	Initial FCS	5	10	32	180	84.11
	Optimized FCS	5	10	17	180	84.11
	C4.5	5	10	17	171	81.42
Seeds	Initial FCS	5	15	243	174	82.86
	Optimized FCS	5	15	12	174	82.86
	C4.5	5	15	12	167	79.52
Wdbc	Initial FCS	10	20	1000	451	79.31
	Optimized FCS	10	20	97	451	79.31
	C4.5	10	20	97	436	76.62
Satimage	Initial FCS	10	20	1000	5246	81.53
	Optimized FCS	10	20	153	5246	81.53
	C4.5	10	20	153	4654	72.32

Table 5: The comparison among the fuzzy classifiers generated by different methods for the four data sets.

Modeling method	Data set	The number of fuzzy rules	The number of the fuzzy rules after optimized	The number of correctly classified samples	Sample size	The classification accuracy (%)
FCS directly established from the samples	Iris	$4^4 = 256$	26	143	150	95.33
	Wine	$2^{13} = 8192$	70	170	178	95.51
	Glass	$2^9 = 512$	118	188	214	87.85
	Seeds	$3^7 = 2187$	34	191	210	90.95
	Wdbc	$3^6 = 729$	102	441	569	77.51
	Satimage	$3^6 = 729$	157	5117	6435	78.31
FCS established and optimized with GA	Iris	$4^2 = 16$	12	142	150	94.67
	Wine	$2^5 = 32$	12	160	178	89.89
	Glass	$2^6 = 64$	17	170	214	79.43
	Seeds	$3^6 = 729$	15	183	210	87.14
	Wdbc	$3^6 = 729$	108	440	569	77.43
	Satimage	$3^6 = 729$	133	5061	6435	78.65
FCS established and optimized with BPSO	Iris	$4^2 = 16$	12	142	150	94.67
	Wine	$2^5 = 32$	12	160	178	89.89
	Glass	$2^6 = 64$	17	173	214	80.84
	Seeds	$3^6 = 729$	12	185	210	88.10
	Wdbc	$3^6 = 729$	104	418	569	73.53
	Satimage	$3^6 = 729$	160	4932	6435	76.65
FCS established and optimized with QiQPSO	Iris	$4^2 = 16$	9	143	150	95.33
	Wine	$2^5 = 32$	10	161	178	90.45
	Glass	$2^5 = 32$	17	180	214	84.11
	Seeds	$3^5 = 243$	12	190	210	90.47
	Wdbc	$3^{10} = 59049$	200	449	569	78.91
	Satimage	$3^{10} = 59049$	153	5189	6435	80.65

fuzzy rules. Although the final fuzzy classifier based on QiQPSO obtained the second highest classification accuracy (84.11%), the number of its fuzzy rules is far smaller, meaning that the classifier has the highest interpretability. Between the GA-based and the BPSO-based classifiers, they have the same number of fuzzy rules, but the latter has higher classification accuracy.

For the Wdbc data set, the fuzzy classification system constructed directly from the samples and improved by GA employs 200 fuzzy rules to obtain a classification of 79.31%.

QiQPSO obtained the final classifier with the second highest classification accuracy (77.51%), which only needs 97 fuzzy rules. It is shown that BPSO had better performance than GA in constructing and optimizing the fuzzy classifier for this data set.

For the Satimage data set, the fuzzy classification system constructed directly from the samples and improved by GA employs 150 fuzzy rules to obtain a classification of 81.53%. QiQPSO yielded the final classifier with the second highest classification accuracy (80.65%), which only needs 12 fuzzy rules. It is shown that BPSO had better performance than GA in constructing and optimizing the fuzzy classifier for this data set.

For the seeds data set, the fuzzy classification system constructed directly from the samples and improved by GA employs 34 fuzzy rules to obtain a classification of 90.95%. QiQPSO obtained the final classifier with the second highest classification accuracy (90.40%), which only needs 153 fuzzy rules. It is shown that BPSO had better performance than GA in constructing and optimizing the fuzzy classifier for this data set.

Overall, from Table 5, we can see that the fuzzy classifiers constructed and optimized by QiQPSO have satisfactory classification accuracies, though not the best ones in general, and has the least number of fuzzy rules. This implies that QiQPSO can generate more interpretable fuzzy classifiers than its competitors with comparable or even better classification accuracies.

6. Conclusion

In this paper, we first proposed the quantum-inspired QPSO (QiQPSO) and then applied the algorithm in optimizing fuzzy rule-based classifiers. Since the original QPSO was developed for continuous optimization problems, we incorporate the quantum computing principles into the QPSO to discretize the particle position and the corresponding operations. In QiQPSO, each particle position is encoded as a string of qubits, the quantum angle of each qubit is updated in line with the equation of QPSO, and a rotation gate is used to update the probability of the qubit according to which the qubit is collapsed to "0" or "1." When QiQPSO is employed to optimize a fuzzy rule-based classifier, the algorithm is first used to construct the initial fuzzy classifier based on the sample data and the grid method of partitioning the feature space. The objectives in construction of the initial fuzzy classification system are to reduce the number of feature variables, minimize the number of fuzzy rules, and maximize the classification accuracy. After the initial classifier is established, its fuzzy rule base is further optimized by QiQPSO in order to minimize the number of fuzzy rules with the classification accuracy unchanged. The proposed method was finally tested on six well-known data sets to construct and optimize the corresponding fuzzy classifiers. The experimental results showed that the QiQPSO can effectively construct the initial fuzzy classifier and optimize its fuzzy rule base. It is also shown that the fuzzy rule-based classifiers optimized by QiQPSO have higher interpretabilities than those optimized by other methods, with comparable and even better classification accuracies.

Data Availability

The data in the manuscript are from the benchmark data set that is accessible on the website.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (project numbers: 61673194, 61672263, and 61672265) and in part by the national first-class discipline program of Light Industry Technology and Engineering (project number: LITE2018-25).

References

[1] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1998.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," vol. 2pp. 192–195, China Machine Press, Beijing, China, 2nd edition, 2004.

[3] K. M. Alexiev and O. I. Georgieva, "Improved fuzzy clustering for identification of takagi-sugeno model," in *Proceedings of the 2nd international IEEE conference on intelligent system*, vol. 1, IEEE, Varna, Bulgaria, Bulgaria, pp. 213–218, June 2004.

[4] H. M. Lee, C. M. Chen, J. M. Chen et al., "An efficient fuzzy classifier with feature selection based on fuzzy entropy," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 3, pp. 426–432, 2001.

[5] Z. Xing, W. Hu, L. Jia, and Q. Chen, "Research on fuzzy modeling based on modified fuzzy clustering and its application," in *Proceedings of the 8th IEEE Conference on Control, Automation, Robotics and Vision*, vol. 3, IEEE, Kunming, China, pp. 2293–2296, December 2004.

[6] V. Snášel, P. Krömer, J. Platoš et al., "The evolution of fuzzy classifier for data mining with applications," *Simulated Evolution and Learning*, pp. 349–358, Springer, Berlin, Germany, 2010.

[7] N. R. Sakthivel, V. Sugumaran, and B. B. Nair, "Automatic rule learning using roughset for fuzzy classifier in fault categorization of mono-block centrifugal pump," *Applied Soft Computing*, vol. 12, no. 1, pp. 196–203, 2012.

[8] A. Lemos, W. Caminhas, and F. Gomide, "Adaptive fault detection and diagnosis using an evolving fuzzy classifier," *Information Sciences*, vol. 220, pp. 64–85, 2013.

[9] T. A. Runkler and R. H. Palm, "Identification of nonlinear systems using regular fuzzy c-elliptotype clustering," in *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, vol. 2, IEEE, New Orleans, LA, USA, pp. 1026–1030, September 1996.

[10] J. M. Dias and A. Dourado, "A self-organizing fuzzy controller with a fixed maximum number of rules and an adaptive similarity factor," *Fuzzy Sets and Systems*, vol. 103, no. 1, pp. 27–48, 1999.

[11] S. Abe and M. S. Ming-Shong Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 18–28, 1995.

[12] G. Castellano and A. M. Fanelli, "Modeling fuzzy classification systems with compact rule base. Computational intelligence for modelling, control & automation: evolutionary computation & fuzzy logic for intelligent control," *Knowledge Acquisition & Information Retrieval*, vol. 2, p. 210, 1999.

[13] M. Valenzuela-Rendón, "The fuzzy classifier system: a classifier system for continuously varying variables," in *Proceedings of the Fourth Intemational Conference on Genetic Algorithms*, Morgan Kauffman), San Diego, CL, USA, July 1991.

[14] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 2, pp. 109–119, 1999.

[15] L. Sánchez, I. Couso, and J. A. Corrales, "Combining GP operators with SA search to evolve fuzzy rule based classifiers," *Information Sciences*, vol. 136, no. 1-4, pp. 175–191, 2001.

[16] H. Ishibuchi and T. Yamamoto, "Effects of three-objective genetic rule selection on the generalization ability of fuzzy rule-based systems," *Evolutionary Multi-Criterion Optimization*, pp. 608–622, Springer, Berlin Germany, 2003.

[17] P. Angelov, X. Zhou, and F. Klawonn, "Evolving fuzzy rule-based classifiers," in *The Proceedings of 2007 Symposium on Computational Intelligence in Image and Signal Processing*, IEEE, Piscataway, NJ, USA, pp. 220–225, April 2007.

[18] C. F. Juang, C. M. Hsiao, and C. H. Hsu, "Hierarchical cluster-based multispecies particle-swarm optimization for fuzzy-system optimization," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 1, pp. 14–26, 2010.

[19] P. Kromer, J. Platos, V. Snasel et al., "Fuzzy classification by evolutionary algorithms," in *The Proceedings of 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, Anchorage, AK, USA, pp. 313–318, October 2011.

[20] E. K. Aydogan, I. Karaoglan, and P. M. Pardalos, "hGA: hybrid genetic algorithm in fuzzy rule-based classification systems for high-dimensional problems," *Applied Soft Computing*, vol. 12, no. 2, pp. 800–806, 2012.

[21] S. García-Galán, R. P. Prado, and J. E. Muñoz Expósito, "Rules discovery in fuzzy classifier systems with PSO for scheduling in grid computational infrastructures," *Applied Soft Computing*, vol. 29, pp. 424–435, 2015.

[22] J. Sun, B. Feng, and W.-B. Xu, "Particle swarm optimization with particles having quantum behavior," in *Procedings of 2004 Congress on Evolutionary Computation*, IEEE, Portland, OR, USA, pp. 325–331, June 2004.

[23] J. Sun, B. Feng, and W.-B. Xu, "A global search strategy of quantum-behaved particle swarm optimization," in *Proceedings of 2004 IEEE Conference on Cybernetics and Intelligent Systems*, IEEE, Singapore, pp. 111–116, December 2004.

[24] J. Sun, W. B. Xu, and B. Feng, "Adaptive parameter control for quantum-behaved particle swarm optimization on individual level," in *Proceedings of 2005 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, Waikoloa, HI, USA, pp. 3049–3054, October 2005.

[25] J. Sun, W. Fang, V. Palade, X. Wu, and W. Xu, "Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point," *Applied Mathematics and Computation*, vol. 218, no. 7, pp. 3763–3775, 2011.

[26] J. Sun, X. Wu, V. Palade, W. Fang, C.-H. Lai, and W. Xu, "Convergence analysis and improvements of quantum-behaved particle swarm optimization," *Information Sciences*, vol. 193, pp. 81–103, 2012.

[27] J. Sun, W. Fang, X. Wu, V. Palade, and W. Xu, "Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection," *Evolutionary Computation*, vol. 20, no. 3, pp. 349–393, 2012b.

[28] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, IEEE, Nagoya, Japan, pp. 61–66, May 1996.

[29] K. H. Han and J. H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 2, IEEE, La Jolla, CA, USA, pp. 1354–1360, July 2000.

[30] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *Evolutionary Computation, IEEE Transactions*, vol. 6, no. 6, pp. 580–593, 2002.

[31] C. Hui, Z. Jiashu, and Z. Chao, "Chaos updating rotated gates quantum-inspired genetic algorithm," in *The Proceedings of the 2004 International Conference on Communications, Circuits and Systems*, vol. 2, IEEE, Chengdu, China, pp. 1108–1112, June 2004.

[32] G. Zhang, W. Jin, and L. Hu, "A novel parallel quantum genetic algorithm," in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, IEEE, Chengdu, China, pp. 693–697, August 2003.

[33] Y. Wang, X.-Y. Feng, Y.-X. Huang et al., "A novel quantum swarm evolutionary algorithm and its applications," *Neurocomputing*, vol. 70, no. 4-6, pp. 633–640, 2007.

[34] H. N. A. Hammed, N. K. Kasabov, and S. M. Shamsuddin, "Quantum-inspired particle swarm optimization for feature selection and parameter optimization in evolving spiking neural networks for classification tasks," *Evolutionary Algorithm*, pp. 133–148, 2011.

[35] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of 1995 IEEE International Conference on Neural Network*, IEEE, Ann Arbor, MI, USA, pp. 1942–1948, December 1995.

[36] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[37] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, IEEE, Orlando, FL, USA, pp. 4104–4141, October 1997.