Soft Comput (2008) 12:567–583
DOI 10.1007/s00500-007-0229-0

ORIGINAL PAPER

An efficient quantum neuro-fuzzy classifier based on fuzzy entropy and compensatory operation

Cheng-Hung Chen · Cheng-Jian Lin · Chin-Teng Lin

Published online: 22 August 2007
© Springer-Verlag 2007

Abstract In this paper, a quantum neuro-fuzzy classifier (QNFC) for classification applications is proposed. The proposed QNFC model is a five-layer structure, which combines the compensatory-based fuzzy reasoning method with the traditional Takagi–Sugeno–Kang (TSK) fuzzy model. The compensatory-based fuzzy reasoning method uses adaptive fuzzy operations of neuro-fuzzy systems that can make the fuzzy logic system more adaptive and effective. Layer 2 of the QNFC model contains quantum membership functions, which are multilevel activation functions. Each quantum membership function is composed of the sum of sigmoid functions shifted by quantum intervals. A self-constructing learning algorithm, which consists of the self-clustering algorithm (SCA), quantum fuzzy entropy and the backpropagation algorithm, is also proposed. The proposed SCA method is a fast, one-pass algorithm that dynamically estimates the number of clusters in an input data space. Quantum fuzzy entropy is employed to evaluate the information on pattern distribution in the pattern space. With this information, we can determine the number of quantum levels. The backpropagation algorithm is used to tune the adjustable parameters. The simulation results have shown that (1) the QNFC model converges quickly; (2) the QNFC model has a higher correct classification rate than other models.

C.-H. Chen · C.-T. Lin
Department of Electrical and Control Engineering,
National Chiao-Tung University,
Hsinchu 300, Taiwan, ROC

C.-J. Lin (✉)
Department of Electrical Engineering,
National University of Kaohsiung,
Kaohsiung 811, Taiwan, ROC
e-mail: cjlin@mail.cyut.edu.tw

Keywords Classification · Compensatory operation · Quantum function · Self-clustering method · Quantum fuzzy entropy · Neuro-fuzzy network

1 Introduction

Classification is one of the most frequent decision-making tasks performed by humans. A classification problem occurs when an object needs to be assigned to a predefined group or class based on the number of observed attributes related to that object. Many problems in business, science, industry, and medicine can be treated as classification problems. Traditional statistical classification procedures, such as discrimination analysis, are built on the Bayesian decision theory (Duda and Hart 1973). In these procedures, an underlying probability model must be assumed in order to calculate the a posteriori probability upon which a classification decision is made. One major limitation of statistical models is that they work well only when the underlying assumptions are correct. The effectiveness of these methods depends to a large extent on the various assumptions or conditions under which the models are developed. Users must have a good knowledge of both data properties and model capabilities before the models can be successfully applied.

Neural networks (Setiono and Liu 1997) have emerged as an important tool for classification tasks. The recent and vast research activities in neural classification have established that neural networks are promising alternatives to various conventional classification methods. However, it is difficult to understand the meaning associated with each neuron and each weight in the neural networks. A fuzzy entropy measure (Lee et al. 2001) is employed to partition the input feature space into decision regions and to select relevant features with good separability for the classification task. However, as compared with the neural networks, learning ability is lock of fuzzy logic. When the views above are summarized, it can be said that, in contrast to pure neural networks or fuzzy systems, the neuro-fuzzy network methods (Halgamuge and Glesner 1994; Kasabov 1996; Nauck and Kruse 1997; Paul and Kumar 2002; Russo 1998; Wang and George Lee 2002) possess the advantages of both neural networks and fuzzy systems. Neuro-fuzzy methods bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

Many papers (Halgamuge and Glesner 1994; Kasabov 1996; Nauck and Kruse 1997; Paul and Kumar 2002; Russo 1998; Wang and George Lee 2002) have dealt with how to optimize fuzzy membership functions and how to choose an optimal defuzzification scheme for applications by using learning algorithms to adjust the parameters of fuzzy membership functions and defuzzification functions. Unfortunately, for optimizing fuzzy logic reasoning and selecting optimal fuzzy operators, only static fuzzy operators are often used to create fuzzy reasoning (Zhang and Kandel 1998). Because the conventional neuro-fuzzy system can only adjust fuzzy membership functions by using fixed fuzzy operations, such as Min and Max, the compensatory neuro-fuzzy system (Zhang and Kandel 1998) with adaptive fuzzy reasoning is more effective and adaptive than the conventional neuro-fuzzy systems with non-adaptive fuzzy reasoning. Therefore, an effective neuro-fuzzy system should be able not only to adaptively adjust fuzzy membership functions but also to dynamically optimize adaptive fuzzy operators.

Recently, quantum neural networks (QNNs) (Purushothaman and Karayiannis 1997; Kretzschmar et al. 2000; Fei et al. 2000) have been developed for detecting and identifying the input data with uncertainty. Conventional NNs and QNNs satisfy the requirements outlined in Leshno et al. (1993) for a universal function approximator. More specifically, QNNs can identify overlaps between data due to their ability to approximate any arbitrary membership profile up to any degree of accuracy. However, QNNs and NNs are generally disadvantaged by their “black box” format, lack a systematic way to determine the appropriate model structure, have no localizability, and converge slowly.

In this paper, a quantum neuro-fuzzy classifier (QNFC) is proposed. The proposed QNFC model includes five major components—quantum membership function, compensatory operation, self-clustering algorithm, quantum fuzzy entropy, and supervised learning method. The proposed QNFC model is a five-layer structure, which combines the compensatory-based fuzzy reasoning method with the traditional Takagi–Sugeno–Kang (TSK) fuzzy model. Layer 2 of the QNFC model contains quantum membership functions, which are multilevel activation functions. Each quantum membership function is composed of the sum of sigmoid functions shifted by quantum intervals. The quantum intervals add an additional degree of freedom that can be exploited during the learning process to capture and quantify the structure of the input space. The advantages of the proposed quantum membership function are summarized as follows: (1) it has the ability of approximate any membership function arbitrarily by the efficient parameters; (2) it has the ability to detect the presence of uncertainty in the input data. At the same time, the compensatory fuzzy inference method uses adaptive fuzzy operations of neuro-fuzzy systems that can make the fuzzy logic system more adaptive and effective.

A self-constructing learning algorithm for the QNFC is also proposed, as follows. First, a structure learning scheme is used to determine proper input space partitioning and to find the center of each cluster. Furthermore, we use quantum fuzzy entropy to determine the number of quantum levels, which reflect the actual distribution of classification patterns. Second, a supervised learning scheme is used to adjust the parameters to obtain the desired outputs. The proposed learning algorithm uses the self-clustering algorithm (SCA), quantum fuzzy entropy to perform structure learning, and the backpropagation algorithm to perform parameter learning. Finally, we evaluate the performance of the proposed QNFC model using two classification problems.

This paper is organized as follows. Section 2 describes the compensatory operation. Section 3 describes the quantum membership function and the structure of the QNFC model. Section 4 describes the learning algorithm of the QNFC model. The self-clustering algorithm, quantum fuzzy entropy, and backpropagation algorithm are presented in this section. In Sect. 5, the QNFC model is used to classify the Iris data and the Wisconsin breast cancer data to demonstrate its learning capability. We also compare our approach with other methods in the literature. Finally, conclusions are given in the last section.

2 The compensatory operation

Zimmermann and Zysno (1980) first defined the essence of compensatory operations. Zhang and Kandel (1998) proposed more extensive compensatory operations based on the pessimistic operation and the optimistic operation. The pessimistic operation can map the inputs x_i to the pessimistic output by making a conservative decision for the pessimistic situation or for even the worst case. For example, $p(x_1, x_2, \ldots, x_n) = MIN(x_1, x_2, \ldots, x_n)$ or $\Pi \quad x_i$. Actually, the t-norm fuzzy operation is a pessimistic operation.

The optimistic operation can map the inputs x_i to the optimistic output by making an optimistic decision for the optimistic situation or for even the best case. For example, $o(x_1, x_2, \ldots, x_n) = MAX(x_1, x_2, \ldots, x_n)$. Actually, the t-conorm fuzzy operation is an optimistic operation. The compensatory operation can map the pessimistic input x_1 and the optimistic input x_2 to make a relatively compromised decision for the situation between the worst case and the best case. For example, $c(x_1, x_2) = x_1^{1-\gamma} x_2^{\gamma}$, where $\gamma \in [0, 1]$ is called the compensatory degree. Many researchers (Ouyang and Lee 1999; Seker et al. 2001; Lin and Ho 2003; Lin and Chen 2003) have used the compensatory operation in fuzzy systems successfully.

The general fuzzy if-then rule is shown as follows:

$$R_j : IF x_1 \ is \ A_{1j} \ and \ldots and \ x_n \ is \ A_{nj} \ THEN \ y \ is \ b_j \quad (1)$$

where x_i and y are the input dimensions and output variables, respectively; A_{ij} is the linguistic term of the precondition part with membership function $\mu_{A_{ij}}$; b_j is the constant consequent; the i is the input dimension, $i = 1, \ldots, n$; the n is the number of existing dimensions; j is the number of rules, $j = 1, \ldots, p$; and p is the number of existing rules.

For an input fuzzy set A' in U, the j th fuzzy rule (1) can generate an output fuzzy set b'_j in v by using the sup-dot composition

$$\mu_{b_j'} = \sup_{\underline{x} \in U} [\mu_{A_{1j} \times \cdots \times A_{nj} \to b_j}(\underline{x}, y) \bullet \mu_{A'}(\underline{x})] \quad (2)$$

where $\underline{x} = (x_1, x_2, \ldots, x_n) . \mu_{A_{1j} \times \cdots \times A_{nj}}(\underline{x})$ is defined in a compensatory operation

$$\mu_{A_{1j} \times \cdots \times A_{nj}}(\underline{x}) = (u_j)^{1-\gamma_j} (v_j)^{\gamma_j} \quad (3)$$

where $\gamma_j \in [0, 1]$ is a compensatory degree. The pessimistic operation and the optimistic operation are as follows:

$$u_j = \prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \quad (4)$$

$$v_j = \left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \right]^{1/n} \quad (5)$$

For simplicity, we can rewrite

$$\mu_{A_{1j} \times \cdots \times A_{nj}}(\underline{x}) = \left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \right]^{1-\gamma_j+\gamma_j/n} \quad (6)$$

Since $\mu_{A'}(x_i) = 1$ for the singleton fuzzifier and $\mu_{b_j'}(y) = 1$, according to (2) we have

$$\mu_{b_j'}(y) = \left[\prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \right]^{1-\gamma_j+\gamma_j/n} \quad (7)$$

Therefore, we can rewrite the fuzzy if-then rule as follows

$$R_j : [IF x_1 \ is \ A_{1j} \ and \ldots and \ x_n is \ A_{nj}]^{1-\gamma_j+\gamma_j/n} \quad THEN \ y \ is \ b_j \quad (8)$$

3 The structure of the QNFC

In this section, we propose the quantum neuro-fuzzy classifier (QNFC). Nodes in layer 1 are input nodes, which represent input variables. Nodes in layer 2 are called quantum membership functions to express the input fuzzy linguistic variables. Nodes in this layer are used to calculate quantum membership values. Each node in layer 3 is called a compensatory rule node. Nodes in layer 3 are equal to the number of compensatory fuzzy sets corresponding to each external linguistic input variable. Links before layer 3 represent the preconditions of the rules, and links after layer 3 represent the consequences of the rule nodes. Nodes in layer 4 are called consequent nodes, where each node is a linear function of the input variables. Nodes in layer 5 are called output nodes, where the node is recommended by layers 3 and 4 and acts as a defuzzifier.

The proposed QNFC realizes a fuzzy if-then rule in the following form (Lin and Chen 2003):

$$Rj : IF \ [x_1 \ is Q_{1j} \ and \ldots and \ x_n \ is Q_{nj}]^{1-\gamma_j+\frac{\gamma_j}{n}} \quad THEN \ y \ is \ a_{0j} + \sum_{i=1}^{n} a_{ij} x_i \quad (9)$$

where x_i and y are the input and output variables, respectively; Q_{ij} is the linguistic term of the precondition part with quantum membership function $\mu_{Q_{ij}}$; $\gamma_j \in [0, 1]$ is a compensatory degree; n is the number of input dimensions; a_{0j} and a_{ij} are the parameters of consequent part; R_j is jth fuzzy rule. The proposed compensatory-based fuzzy reasoning method is based on our previous research (Lin and Chen 2003).

The membership function of the precondition part discussed in this paper is different from the typical Gaussian membership function. We propose the quantum membership function to approximate desired results. The basic concept of quantum membership function is derived from a multilevel transfer function (Purushothaman and Karayiannis 1997). Therefore, the response of the jth quantum membership function for the ith feature vector can be written as (Lin et al. 2004)

$$Q_{ij} = \frac{1}{ns_{ij}} \sum_{\alpha=1}^{ns_{ij}} \left[\left(\frac{1}{1 + \exp(-\beta(x_i - m_{ij} + |\theta_{ij}^{\alpha}|))} \right) \times U\left(x_i; -\infty, m_{ij}\right) \right.$$

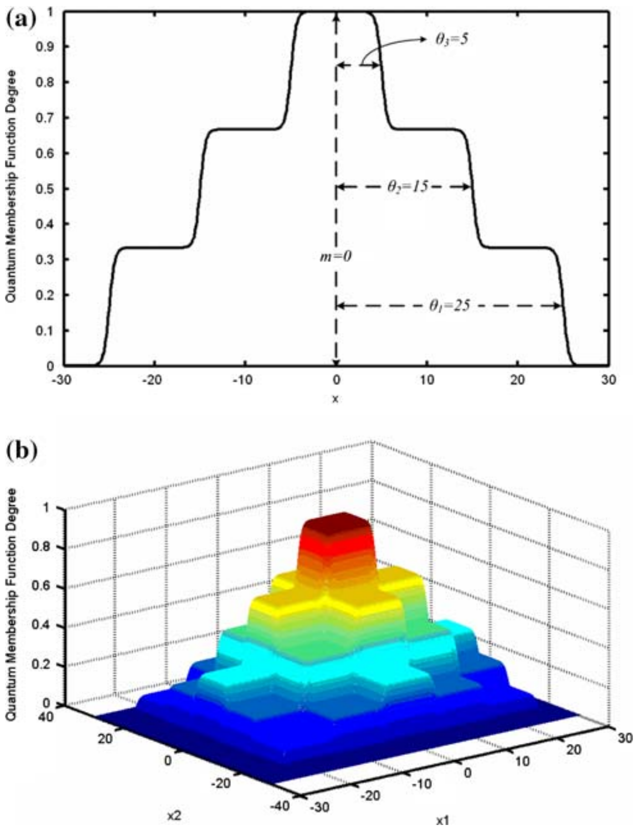


Fig. 1 Quantum membership function shown in **a** one-dimension and **b** two-dimensions

$$+\left(\frac{\exp(-\beta(x_i - m_{ij} - |\theta_{ij}^{\alpha}|))}{1+\exp(-\beta(x_i - m_{ij} - |\theta_{ij}^{\alpha}|))}\right)$$
$$\times U\left(x_i; m_{ij}, \infty\right)\Bigg] \tag{10}$$

where $U(x_i; a, b) = \begin{cases} 1, & \text{if } a \le x_i < b \\ 0, & \text{otherwise} \end{cases}$, β is the slope factor, θ_{ij}^{α} is the quantum interval, m_{ij} is the center of the quantum membership function, and ns_{ij} is the number of levels in the quantum membership function for the jth rule of the ith input. Therefore, we can describe the fuzzy if-then rule as follows:

$$Rj : IF\Big[x_1 is\ \mu\left(m_{1j}; \theta_{1j}^{\alpha}\right) and \ldots and x_i\ is\ \mu\left(m_{ij}; \theta_{ij}^{\alpha}\right)$$
$$and \ldots and\ x_n\ is\ \mu\left(m_{nj}; \theta_{nj}^{\alpha}\right)\Big]^{1-\gamma_j+\frac{\gamma_j}{n}}$$
$$THEN\ y\ is\ a_{0j} + a_{1j}x_1 \ldots + a_{ij}x_i + \cdots + a_{nj}x_n \tag{11}$$

Figure 1 shows the response of a three-level quantum membership function using Eq. (10) with $\beta = 1, \theta = [5, 15, 25]$, $m = 0, ns = 3$.

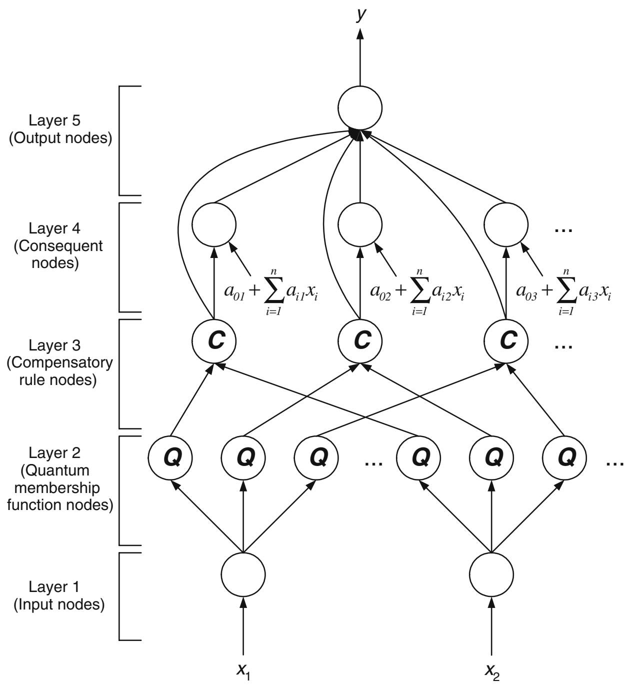

y
Layer 5 (Output nodes)
Layer 4 (Consequent nodes)
Layer 3 (Compensatory rule nodes)
Layer 2 (Quantum membership function nodes)
Layer 1 (Input nodes)
$a_{01}+\sum_{i=1}^{n}a_{i1}x_i$
$a_{02}+\sum_{i=1}^{n}a_{i2}x_i$
$a_{03}+\sum_{i=1}^{n}a_{i3}x_i$
C
Q
...
x_1
x_2

Fig. 2 Structure of the proposed QNFC

The structure of the quantum neuro-fuzzy classifier (QNFC), which is systematized into n input variables, p-term nodes for each input variable, one output node, and $n \times p$ membership function nodes, is shown in Fig. 2. We shall introduce the operation functions of the nodes in each layer of the QNFC model. In the following description, $u^{(l)}$ denotes an output of a node in the lth layer.

Layer 1 (*input node*): No computation is done in this layer. Each node in this layer is an input node, which corresponds to one input variable and which only transmits input values to the next layer directly.

$$u_i^{(1)} = x_i \tag{12}$$

Layer 2 (*quantum membership function node*): Nodes in this layer correspond to one linguistic label of the input variables in layer1 and a unit of memory. That is, the membership value specifying the degree to which an input value and a unit of memory belong to a fuzzy set is calculated in layer 2. The quantum membership function (Lin et al. 2004), the operation performed in layer 2 is

$$u_{ij}^{(2)} = \frac{1}{ns_{ij}} \sum_{\alpha=1}^{ns_{ij}} \Bigg[\left(\frac{1}{1+\exp(-\beta(u_i^{(1)} - m_{ij} + |\theta_{ij}^{\alpha}|))}\right)$$
$$\times U\left(u_i^{(1)}; -\infty, m_{ij}\right)$$

$$+\left(\frac{\exp(-\beta(u_i^{(1)} - m_{ij} - |\theta_{ij}^{\alpha}|))}{1+\exp(-\beta(u_i^{(1)} - m_{ij} - |\theta_{ij}^{\alpha}|))}\right)$$
$$\times U\left(u_i^{(1)}; m_{ij}, \infty\right)\Bigg] \quad (13)$$

where $U(x_i; a, b) = \begin{cases} 1, & \text{if} a \le x_i < b \\ 0, & \text{otherwise} \end{cases}$, β is the slope factor, θ_{ij}^{α} is the quantum interval, m_{ij} is the center of the quantum membership function, and ns_{ij} is the number of levels in the quantum membership function for the jth rule of the ith input.

Layer 3 (*compensatory rule node*): Nodes in this layer represent the preconditioned part of one fuzzy logic rule. They receive one-dimensional membership degrees of the associated rule from nodes of a set in layer 2. Here, we use the compensatory operator previously mentioned to perform IF-condition matching of fuzzy rules. As a result, the output function of each inference node is (Lin and Chen 2003)

$$u_j^{(3)} = \left(\prod_i u_{ij}^{(2)}\right)^{1-\gamma_j+\frac{\gamma_j}{n}} \quad (14)$$

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule, and $\gamma_j = c_j^2 \Big/ \left(c_j^2 + d_j^2\right) \in [0, 1]$ is called the compensatory degree and $c_j, d_j \in [-1, 1]$. The purpose of tuning c_j and d_j is to increase the adaptability of the fuzzy operator.

Layer 4 (*consequent node*): Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output delivered from layer 3, and the other inputs are the input variables from layer 1 as depicted in Fig. 2. For this kind of node, we have

$$u_j^{(4)} = u_j^{(3)}(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i) \quad (15)$$

where the summation is over all the inputs and where a_{ij} are the corresponding parameters of the consequent part.

Layer 5 (*output node*): Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{p} u_j^{(4)}}{\sum_{j=1}^{p} u_j^{(3)}} = \frac{\sum_{j=1}^{p} u_j^{(3)}(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i)}{\sum_{j=1}^{p} u_j^{(3)}} \quad (16)$$

where p is the number of fuzzy rule.

4 A learning algorithm for the QNFC model

In this section, we present a learning algorithm for the proposed QNFC model. The following two schemes are part of this learning algorithm. First, a structure learning scheme is proposed to determine proper input space partitioning and to find the center of each cluster. Furthermore, we propose quantum fuzzy entropy to decide the number of quantum levels that reflect the actual distribution of classification patterns. Second, a supervised learning scheme is proposed to adjust the parameters for the desired outputs. The proposed learning algorithm uses the self-clustering algorithm (SCA), quantum fuzzy entropy to perform structure learning and the backpropagation algorithm to perform parameter learning.

4.1 The structure learning

The first step in structure learning is to determine the number of rules using the self-clustering algorithm (SCA) from the training data, as well as to determine the number of fuzzy sets in the universe of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, with m_{ij} and θ_{ij}^{α} representing the center and the quantum interval, respectively. Simultaneously, we employ quantum fuzzy entropy to determine the appropriate number of quantum levels. After the self-clustering algorithm (SCA), the quantum intervals and the number of quantum levels are determined, it is then easy to decide on the quantum membership function.

4.1.1 The self-clustering algorithm

Layer 2 of the QNFC model can be viewed as a function that maps input patterns. Hence, the discriminative ability of these new features is determined by the centers of the quantum membership function. To achieve good classification, centers are best selected based on their ability to provide large class separation.

A clustering method, called the self-clustering algorithm (SCA), is proposed to implement scatter partitioning of the input space. Without any optimization, the online SCA is a fast, one-pass algorithm for a dynamic estimation of the number of clusters in a set of data and for finding the current centers of clusters in the input data space. It is a distance-based connectionist-clustering algorithm. In any cluster, the maximum distance between a sample point and the cluster center is less than a threshold value which has been set as a clustering parameter and which would affect the number of clusters to be estimated.

In the clustering process, the data samples come from a data stream. The process starts with an empty set of clusters. When a new cluster is created, the cluster center, C, is defined, and its cluster distance and cluster width, Dc and

Wd, is initially set to zero. When more samples are presented one after another, some created clusters will be updated by changing the positions of their centers and increasing the cluster distances and cluster width. Which cluster will be updated and how much it will be changed depends on the position of the current sample in the input space. A cluster will not be updated any more when its cluster distance, Dc, reaches the value that is equal to the threshold value D_{thr}. In the clustering process, the threshold parameter D_{thr} is an important parameter. A low threshold value leads to the learning of coarse clusters (i.e., less rules are generated), whereas a high threshold value leads to the learning of fine clusters (i.e., more rules are generated). Therefore, the selection of the threshold value D_{thr} will critically affect the simulation results, and the value will be based on practical experimentation or on trial-and-error tests. We defined generally that D_{thr} is equal to 0.5–1 times of summation of the samples variance.

Figure 3 briefly shows the SCA clustering process in two-input space. The SCA is described as follows.

Step 1: We have to disarrange the order of the original data samples by randomization. Create the first cluster by simply taking the position of the first sample from the input stream as the first cluster center C_1, and setting its cluster distance Dc_1 and cluster width Wd_1_x and Wd_1_y to zero, as shown in Fig. 3(a).

Step 2: If all samples of the data stream have been processed, the algorithm is finished. Otherwise, the current input sample, P_i, is taken and the distances between this sample and all p already created cluster centers C_j, $Dist_{ij} = \|P_i - C_j\|$, $j = 1, 2, \ldots, p$, are calculated.

Step 3: If there is any distance value $Dist_{ij}$ equal to, or less than, at least one of the distance Dc_j, $j = 1, 2, \ldots, p$, it means that the current sample P_i belongs to a cluster C_m with the minimum distance

$$Dist_{im} = \|P_i - C_m\| = \min\left(\|P_i - C_j\|\right), \quad j = 1, 2, \ldots, p \tag{17}$$

In this case, neither a new cluster is created, nor any existing cluster is updated, as in the cases of P_4 and P_6 shown in Fig. 3, for example. The algorithm then returns to Step2. Otherwise, the algorithm goes to the next step.

Step 4: Find a cluster with center C_m and cluster distance Dc_m from all n existing cluster centers by calculating the values $S_{ij} = Wd_{ij}+Dc_j$, $j = 1, 2, \ldots, p$, and then choosing the cluster center C_m with the minimum value S_{im}:

$$S_{im} = Wd_{im} + Dc_m = \min(S_{ij}), \quad j = 1, 2, \ldots, p \tag{18}$$

In Eq. (17), the maximum distance from any cluster center to the samples that belong to this cluster is not greater than the threshold, D_{thr}, though the algorithm does not keep any information of passed samples. However, we find that the formulation only considers the distance between the input data and cluster center in Eq. (18). But the special situation

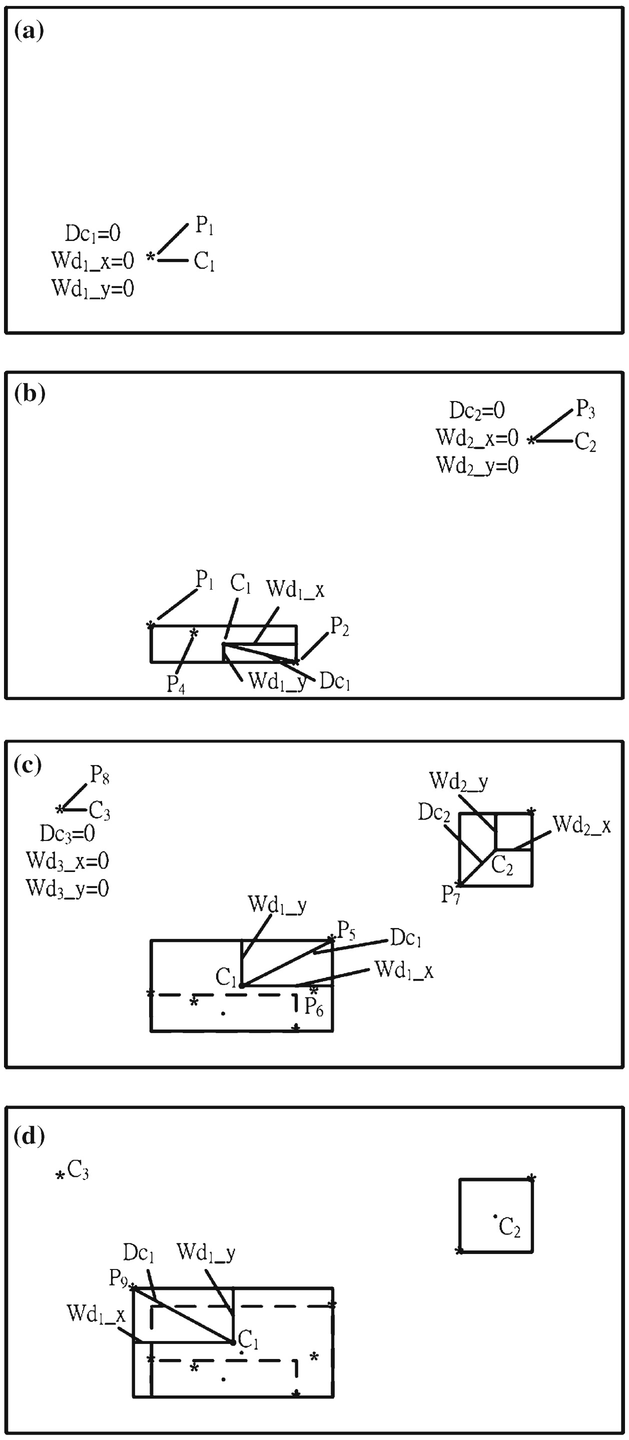


Fig. 3 A brief clustering process using the SCA with samples P_1 to P_9 in 2D space. **a** The sample P_1 causes the SCA to create a new cluster center C_1. **b** P_2: update cluster center C_1, P_3: create a new cluster center C_2, P_4: do nothing. **c** P_5: update cluster C_1, P_6: do nothing, P_7: update cluster center C_2, P_8: create a new cluster C_3. **d** P_9: update cluster C_1

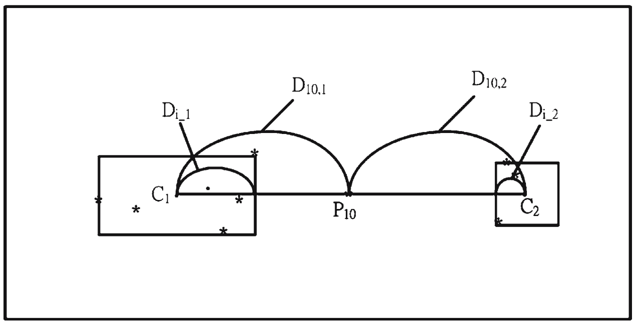


Fig. 4 The special case of SCA

shows that the distances between a given point P_{11} and both cluster centers Dc_1 and Dc_2 are the same as shown in Fig. 4. In the aforementioned technique, the cluster C_2, which has small dimension distances D_{2_x}, will be selected to expand according to Eq. (18). However, this causes a problem in that the cluster numbers increase quickly. To avoid this problem, we make a judgment, as follows:

If (*the distance between* P_{11} *and* Dc_1 *is equal to the distance between* P_{11} *and* Dc_2) *and* ($D_{1\_x} > D_{2\_x}$)

then $D_{im} = Dc_1$

From the above rule, we find that when the distances between the input data and both clusters are the same, the formulation will choose the cluster that has large dimension distances D_{1_x}.

Step 5: If S_{im} is greater than D_{thr}, the sample P_i does not belong to any existing clusters. A new cluster is created in the same way as described in Step 1, as in the cases of P_3 and P_8 shown in Fig. 3, and the algorithm returns to Step 2.

Step 6: If S_{im} is not greater than D_{thr}, the cluster C_m is updated by moving its center, C_m, and increasing the value of its cluster distance, Dc_m, and cluster width Wd_m_x, Wd_m_y. The parameters are updated by the following equation:

$$Wd_m\_x^{new} = (\|C_m\_x - P_i\_x\| + Wd_m\_x)/2 \quad (19)$$

$$Wd_m\_y^{new} = (\|C_m\_y - P_i\_y\| + Wd_m\_y)/2 \quad (20)$$

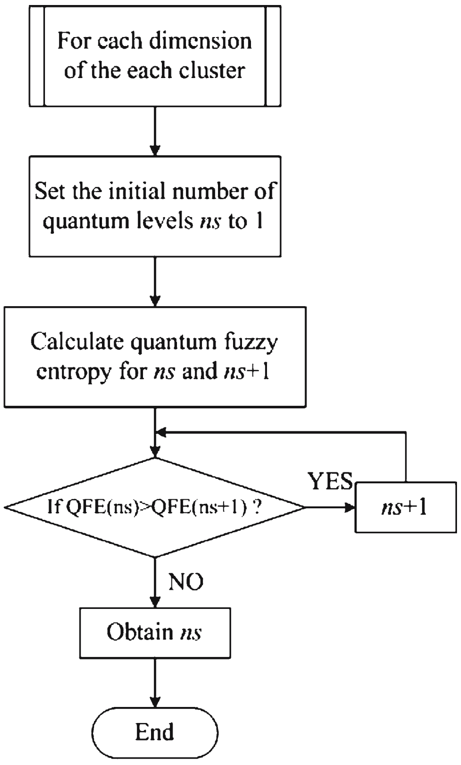
$$C_m\_x^{new} = P_i\_x - D_m\_x^{new} \quad (21)$$

$$C_m\_y^{new} = P_i\_y - D_m\_y^{new} \quad (22)$$

$$Dc_m^{new} = S_{im}/2 \quad (23)$$

where C_m_x is the value of the x dimension for C_m, C_m_y is the value of the y dimension for C_m, P_i_x is the value of the x dimension for P_i, and P_i_y is the value of the y dimension for P_i, as in the cases of P_2, P_5, P_7, and P_9 shown in Fig. 3. The algorithm returns to Step 2.

In this way, the maximum distance from any cluster center to the samples that belong to this cluster is not greater than the threshold value D_{thr}, though the algorithm does not keep any information of passed samples. After that, the center and the quantum interval of the quantum membership function



For each dimension of the each cluster
Set the initial number of quantum levels *ns* to 1
Calculate quantum fuzzy entropy for *ns* and *ns*+1
If QFE(ns)>QFE(ns+1) ?
YES
ns+1
NO
Obtain *ns*
End

Fig. 5 The flow diagram of the determination of quantum level number

are defined by the following equation:

$$m_{ij} = C_j, \quad j = 1, 2, \ldots, p \quad (24)$$

$$\theta_{ij}^{\alpha} = \frac{1}{((ns_{ij}+1)/2)} \cdot \alpha \cdot D_j, \quad \alpha = 1, 2, \ldots, ns_{ij}, \quad j = 1, 2, \ldots, p \quad (25)$$

4.1.2 Quantum fuzzy entropy

After that, the center and the quantum interval of the quantum membership function are determined. The number of quantum levels in each dimension has a profound effect on learning efficiency and classification accuracy. If the number of quantum levels is too large, it will take too long to finish the training and classification processes, and overfitting may result. On the other hand, if the number of quantum levels is too small, the size of each decision region may be too big to fit the distribution of input patterns, and classification performance may suffer.

Therefore, the selection of the optimal number of quantum levels is an important task. In this subsection, we will investigate a systematic method to select the appropriate number of quantum levels. The flow diagram of the determination of quantum level number using quantum fuzzy entropy is shown in Fig. 5. The proposed criterion is based on quantum fuzzy entropy, since it has the ability to reflect the actual distribution of pattern space. Figure 6 briefly shows the clustering

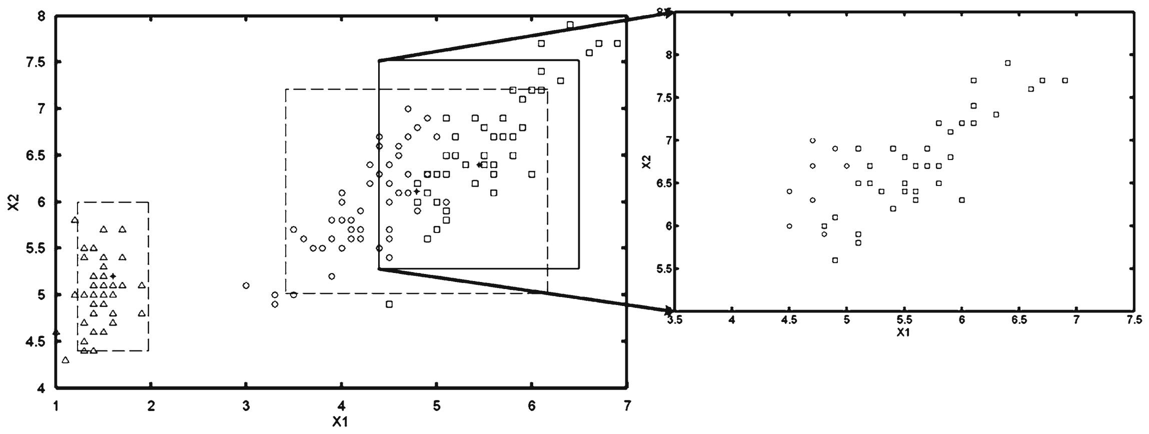


Fig. 6 The pattern distribution with two dimensions and three classes of the cluster

results of pattern distribution after the SCA clustering process in two-input space. The steps involved in selecting the quantum level number for each dimension of the each cluster are described as follows:

Step 1: Set the initial number of quantum levels *ns* to 1, i.e. the number of quantum levels is equal to one.

Step 2: Locate the centers and the quantum intervals. The self-clustering algorithm will be used to locate the center and the quantum interval of each cluster.

Step 3: Assign a quantum membership function to each cluster. In order to apply quantum fuzzy entropy to calculate the distribution information of patterns in a cluster, we have to assign a quantum membership function to each cluster.

Step 4: Compute the total quantum fuzzy entropy for all clusters in each dimension for $ns = 1$ and $ns = 2$. We compute the quantum fuzzy entropy for all clusters in each dimension to obtain the distribution information of patterns projected in this dimension. The notations in the quantum fuzzy entropy are described as follows:

X	all elements are the same cluster
$\tilde{Q}$	the quantum fuzzy set with *ns* quantum levels
$\mu_{\tilde{Q}}(x_i)$	the degree of the element x_i with the quantum fuzzy set $\tilde{Q}$
p	the number of classification
C_j	the jth class for classification C
$T_{C_j}(x_n)$	the set of elements of class j in the cluster X
SD_j	the calculated degree with the quantum fuzzy set $\tilde{Q}$ for the elements of class j in the quantum levels *ns*
$QFE_{C_j}(\tilde{Q})$	the quantum fuzzy entropy for the elements of class j in the quantum levels *ns*
$QFE(\tilde{Q})$	the quantum fuzzy entropy for the elements within the quantum levels *ns* in the cluster X

Quantum fuzzy entropy is defined as follows:

(1) Let $X = \{x_1, x_2, \ldots, x_n\}$ be a cluster set with elements x_i distributed in a pattern space, where $i = 1; 2; \ldots; n$.
(2) Let $\tilde{Q}$ be a quantum fuzzy set defined in the quantum levels *ns* of a pattern space. The mapped quantum membership degree of the element x_i with the quantum fuzzy set $\tilde{Q}$ is denoted by $\mu_{\tilde{Q}}(x_i)$.
(3) Let $C_1; C_2; \ldots; C_p$ represent p classes into which the n elements are divided. (e.g. there are two classes ○ and □ in Fig. 6)
(4) Let $T_{C_j}(x_n)$ denote a set of elements of class j in the cluster X. It is a subset of the cluster X.
(5) The sub-degree SD_j with the quantum fuzzy set $\tilde{Q}$ for the elements of class j in the quantum levels *ns*, where $j = 1; 2; \ldots; p$, is defined as

$$SD_j = \frac{\sum_{x \in T_{C_j}(x_n)} \mu_{\tilde{Q}}(x)}{\sum_{x \in X} \mu_{\tilde{Q}}(x)} \tag{26}$$

(6) The quantum fuzzy entropy $QFE_{C_j}(\tilde{Q})$ of the elements of class j in the quantum levels *ns* is defined as (Lee et al. 2001)

$$QFE_{C_j}(\tilde{Q}) = -SD_j \log SD_j \tag{27}$$

(7) The quantum fuzzy entropy $QFE(\tilde{Q})$ for the elements within the quantum levels *ns* in the cluster X is defined as

$$QFE(\tilde{Q}) = \sum_{j=1}^{p} QFE_{C_j}(\tilde{Q}) \tag{28}$$

In this step, we can compute the quantum fuzzy entropy for the quantum levels $ns = 1$ and $ns = 2$, as shown in Fig. 7.

Step 5: If the total quantum fuzzy entropy of $ns + 1$ quantum levels is less than that of *ns* quantum levels, then $ns = ns + 1$. Then go to *Step 2*. Otherwise, else go to *Step 6*.

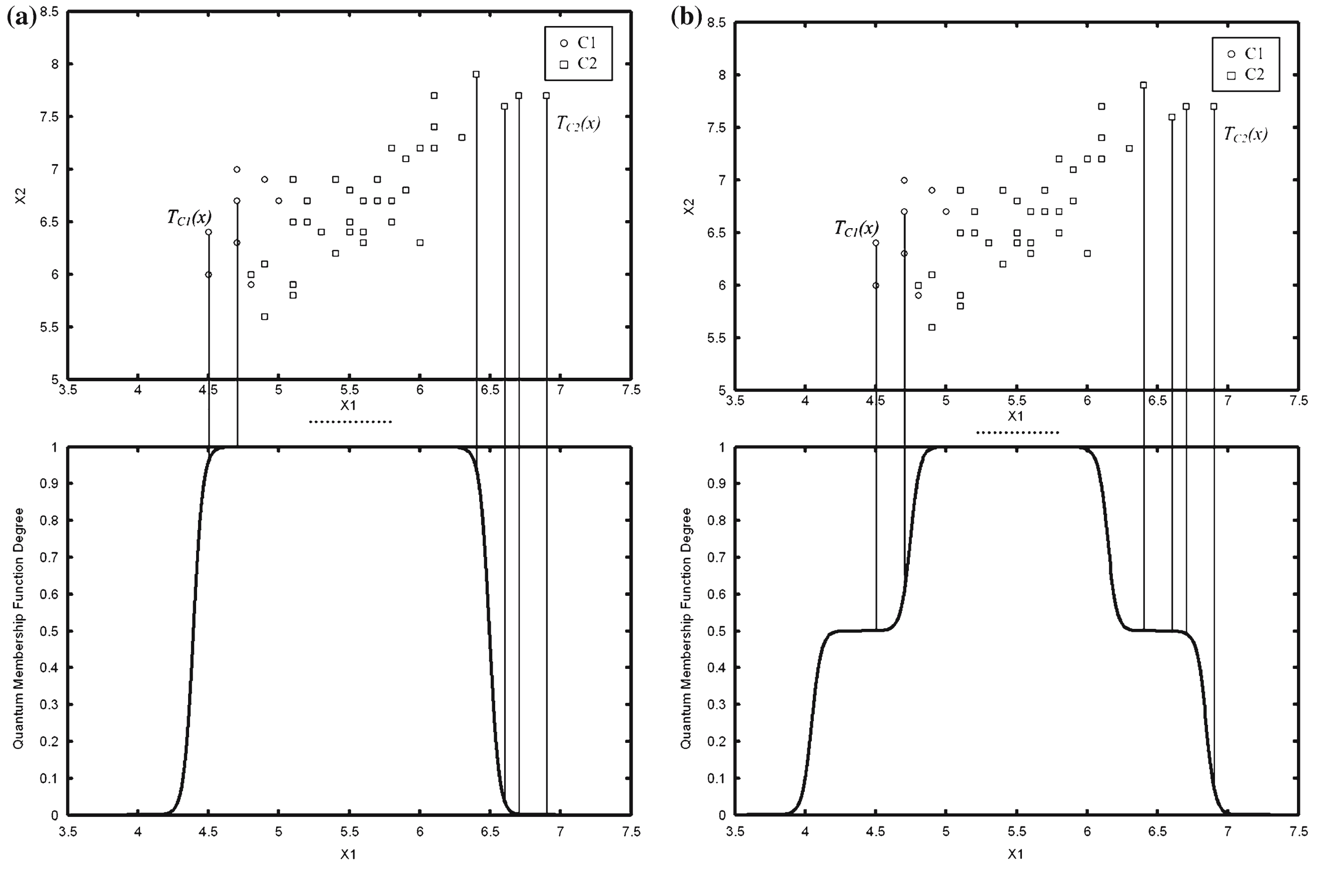


Fig. 7 The pattern distribution with corresponding quantum membership function. **a** The number of quantum levels is one and **b** the number of quantum levels is two

Step 6: The term *ns* represents the number of quantum levels in a specified dimension. Since the quantum fuzzy entropy does not decrease, we stop increasing the quantum level in this dimension, and we let *ns* be the number of quantum levels in this dimension.

4.2 The parameter learning

After the network structure is determined by the self-clustering algorithm, the network then enters the parameter learning phase to adjust the parameters of the network based on the training patterns. The learning process involves minimizing a given cost function. The gradient of the cost function is computed and adjusted along the negative gradient. The backpagation algorithm (Lin et al. 2004) is used for this supervised learning method. When we consider the single output case for clarity, our goal to minimize the cost function E is defined as

$$E = \frac{1}{2}[y - y^d]^2 \tag{29}$$

where y^d is the desired output and y is the current output. Then the parameter learning algorithm based on backpropagation is described as follows:

The error term to be propagated is calculated as

$$\delta_e = -\frac{\partial E}{\partial y} = y^d - y \tag{30}$$

The parameter of consequent part is updated by the amount

$$\Delta a_{0j} = -\frac{\partial E}{\partial a_{0j}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial u_j^{(4)}}\right]\left[\frac{\partial u_j^{(4)}}{\partial a_{0j}}\right] = \frac{\delta_e u_j^{(3)}}{\sum_{j=1}^{p} u_j^{(3)}} \tag{31}$$

and

$$\Delta a_{ij} = -\frac{\partial E}{\partial a_{ij}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial u_j^{(4)}}\right]\left[\frac{\partial u_j^{(4)}}{\partial a_{ij}}\right] = \frac{\delta_e u_j^{(3)} x_i}{\sum_{j=1}^{p} u_j^{(3)}} \tag{32}$$

The parameter of consequent part in the output layer is updated according to the following equation:

$$a_{0j}(t+1) = a_{0j}(t) + \eta_a \Delta a_{0j} \tag{33}$$
$$a_{ij}(t+1) = a_{ij}(t) + \eta_a \Delta a_{ij} \tag{34}$$

where factor η_a is the learning rate parameter of the parameter and t denotes the jth iteration number . The output error (i.e., the difference between the desired output and the current output) is then backpropagated to update their compensatory degree, centers and quantum intervals. According to the chain rule, the updated compensatory degree is as follows:

$$\begin{aligned}\Delta\gamma_j &= -\frac{\partial E}{\partial \gamma_j} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial \gamma_j}\right]\\ &= \delta_e \cdot \left[\frac{\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\cdot \sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}\left(u_j^{(3)}\cdot\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\right)}{\left(\sum_{j=1}^{p} u_j^{(3)}\right)^2}\right]\cdot\left[\frac{1}{n}-1\right]\cdot \ln\left[\prod_i u_{ij}^{(2)}\right]\cdot\left[u_j^{(3)}\right] \qquad (35)\end{aligned}$$

We have

$$c_j(t+1) = c_j(t) + \eta_c\left\{\frac{2c_j(t)d_j^2(t)}{[c_j^2(t)+d_j^2(t)]^2}\right\}\Delta\gamma_j \qquad (36)$$

$$d_j(t+1) = d_j(t) - \eta_d\left\{\frac{2c_j^2(t)d_j(t)}{[c_j^2(t)+d_j^2(t)]^2}\right\}\Delta\gamma_j \qquad (37)$$

$$\gamma_j(t+1) = \frac{c_j^2(t+1)}{c_j^2(t+1)+d_j^2(t+1)} \qquad (38)$$

In Eqs. (36) and (37), η_c and η_d are the learning rate of the parameter c_j and the parameter d_j, respectively. The parameters c_j and d_j are factors of the compensatory degree γ_j. The quantum function memberships of layer 2 are updated for their centers and quantum intervals. The updated center is as follows:

$$\begin{aligned}\Delta m_{ij} &= -\frac{\partial E}{\partial m_{ij}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial m_{ij}}\right]\\ &= \delta_e\cdot\left[\frac{\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\cdot\sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}\left(u_j^{(3)}\cdot\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\right)}{\left(\sum_{j=1}^{p} u_j^{(3)}\right)^2}\right]\\ &\quad\times\left[1-\gamma_j+\frac{\gamma_j}{n}\right]\cdot\left[\prod_i u_{ij}^{(2)}\right]^{-\gamma_j+\frac{\gamma_j}{n}}\cdot\left[\prod_{\substack{j=1\\ i\neq j}}^{p} u_{ij}^{(2)}\right]\\ &\quad\times\frac{1}{ns_{ij}}\sum_{\alpha=1}^{ns_{ij}}\left[-\frac{\beta\cdot\left(\exp\left(-\beta\cdot\left(x_i-m_{ij}+\left|\theta_{ij}^{\alpha}\right|\right)\right)\right)}{\left(1+\exp\left(-\beta\cdot\left(x_i-m_{ij}+\left|\theta_{ij}^{\alpha}\right|\right)\right)\right)^2}\cup(x_i;-\infty,m_{ij})\right.\\ &\quad\left.+\frac{\beta\cdot\left(\exp\left(-\beta\cdot\left(x_i-m_{ij}+\left|\theta_{ij}^{\alpha}\right|\right)\right)\right)}{\left(1+\exp\left(-\beta\cdot\left(x_i-m_{ij}+\left|\theta_{ij}^{\alpha}\right|\right)\right)\right)^2}\cup(x_i;m_{ij},\infty)\right] \qquad (39)\end{aligned}$$

The updated quantum interval is as follows:
If $\theta_{ij}^{\alpha}\geq 0$, then

$$\begin{aligned}\Delta\theta_{ij}^{\alpha} &= -\frac{\partial E}{\partial\theta_{ij}^{\alpha}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial\theta_{ij}^{\alpha}}\right]\\ &= \delta_e\cdot\left[\frac{\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\cdot\sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}\left(u_j^{(3)}\cdot\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\right)}{\left(\sum_{j=1}^{p} u_j^{(3)}\right)^2}\right]\\ &\quad\times\left[1-\gamma_j+\frac{\gamma_j}{n}\right]\cdot\left[\prod_i u_{ij}^{(2)}\right]^{-\gamma_j+\frac{\gamma_j}{n}}\cdot\left[\prod_{\substack{j=1\\ i\neq j}}^{p} u_{ij}^{(2)}\right]\\ &\quad\times\frac{1}{ns_{ij}}\cdot\left[\frac{\beta\cdot\left(\exp\left(-\beta\cdot\left(x_i-m_{ij}+\theta_{ij}^{\alpha}\right)\right)\right)}{\left(1+\exp\left(-\beta\cdot\left(x_i-m_{ij}+\theta_{ij}^{\alpha}\right)\right)\right)^2}\cup(x_i;-\infty,m_{ij})\right.\\ &\quad\left.-\frac{\beta\cdot\left(\exp\left(-\beta\cdot\left(x_i-m_{ij}+\theta_{ij}^{\alpha}\right)\right)\right)}{\left(1+\exp\left(-\beta\cdot\left(x_i-m_{ij}+\theta_{ij}^{\alpha}\right)\right)\right)^2}\cup(x_i;m_{ij},\infty)\right] \qquad (40)\end{aligned}$$

else $\theta_{ij}^{\alpha} < 0$

$$\begin{aligned}\Delta\theta_{ij}^{\alpha} &= -\frac{\partial E}{\partial\theta_{ij}^{\alpha}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial\theta_{ij}^{\alpha}}\right]\\ &= \delta_e\cdot\left[\frac{\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\cdot\sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}\left(u_j^{(3)}\cdot\left(a_{0j}+\sum_{i=1}^{n} a_{ij}x_i\right)\right)}{\left(\sum_{j=1}^{p} u_j^{(3)}\right)^2}\right]\\ &\quad\times\left[1-\gamma_j+\frac{\gamma_j}{n}\right]\cdot\left[\prod_i u_{ij}^{(2)}\right]^{-\gamma_j+\frac{\gamma_j}{n}}\cdot\left[\prod_{\substack{j=1\\ i\neq j}}^{p} u_{ij}^{(2)}\right]\\ &\quad\times\frac{1}{ns_{ij}}\cdot\left[-\frac{\beta\cdot\left(\exp\left(-\beta\cdot\left(x_i-m_{ij}-\theta_{ij}^{\alpha}\right)\right)\right)}{\left(1+\exp\left(-\beta\cdot\left(x_i-m_{ij}-\theta_{ij}^{\alpha}\right)\right)\right)^2}\cup(x_i;-\infty,m_{ij})\right.\\ &\quad\left.+\frac{\beta\cdot\left(\exp\left(-\beta\cdot\left(x_i-m_{ij}-\theta_{ij}^{\alpha}\right)\right)\right)}{\left(1+\exp\left(-\beta\cdot\left(x_i-m_{ij}-\theta_{ij}^{\alpha}\right)\right)\right)^2}\cup(x_i;m_{ij},\infty)\right] \qquad (41)\end{aligned}$$

The centers and quantum intervals of the quantum function neurons in this layer are updated as follows:

$$m_{ij}(t+1) = m_{ij}(t) + \eta_m\Delta m_{ij} \qquad (42)$$

$$\theta_{ij}^{\alpha}(t+1) = \theta_{ij}^{\alpha}(t) + \eta_\theta\Delta\theta_{ij}^{\alpha} \qquad (43)$$

where η_m and η_θ are the learning rate parameters of the center and the quantum interval of the quantum function neurons, respectively.

5 Illustrative examples

In this section, we evaluate the performance of the proposed QNFC model using two better-known benchmark data sets used for classification problems. The first example uses the Iris data and the second example uses the Wisconsin breast cancer data. The two benchmark data sets are available from the University of California, Irvine, via an anonymous ftp address ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

In the following simulations, the parameters and number of training epochs were based on the desired accuracy. In short, the trained QNFC model was stopped once its high learning efficiency was demonstrated.

Example 1: Iris data classification

The Fisher–Anderson Iris data consists of four input measurements, sepal length (*sl*), sepal width (*sw*), petal length (*pl*), and petal width (*pw*), on 150 specimens of the Iris plant. Three species of Iris were involved, *Iris sestosa*, *Iris versiolor* and *Iris virginica*, and each species contains 50 instances. The measurements are shown in Fig. 8.

In the Iris data experiment, 25 instances with four features from each species were randomly selected as the training set (i.e., a total of 75 training patterns were used as the training data set) and the remaining instances were used as the testing set. The 75 training patterns were obtained via a random selection process from the original Iris dataset of 150 patterns. For the self-clustering algorithm (SCA), we chose the parameter $D_{thr} = 4.5$. Furthermore, we determined the different number of quantum levels for each dimension of each cluster using quantum fuzzy entropy and tabulated them in Table 1. After structure learning, three clusters were generated.

The network then entered the parameter learning phase. We set the learning rate to $\eta = 0.01$ and trained the QNFC model with different quantum levels for each dimension of each cluster. After 100 training steps, the final rms error was 0.0133. Three fuzzy logic rules were generated. The three designed fuzzy rules were

Figures 9a–f show the distribution of the training patterns and the final assignment of the fuzzy rules (i.e., the distribution of the input membership functions). The boundary of each rectangle represents a rule with a firing strength of 0.5. Figure 10 shows the learned QNFC structure and the corresponding quantum membership functions for each dimension after the parameter learning phase. We compared the testing accuracy of our model with that of other methods—the traditional multilayer neural network, the standard radial basis function network (RBFN) with the self-clustering algorithm (SCA), and the QNFC without compensatory operation. Five experiments were used. These experiments calculated the classification accuracy and the values of the average produced on the testing set using the traditional multilayer neural network, the radial basis function network (RBFN) with the self-clustering algorithm (SCA), the QNFC model without compensatory operation, and the proposed QNFC model.

During the learning phase, the learning curves from the proposed QNFC model, the QNFC without compensatory operation, and the RBFN with the SCA model are shown in Fig. 11. Table 2 shows that the experiments with the QNFC model result in high accuracy, with an accuracy percentage ranging from 96 to 98.67%. The means of re-substitution accuracy was 97.6%. The average classification accuracy of the QNFC model was better than that of other methods. In

Rule 1: IF [*sl* is μ(4.94;0.86) and *sw* is μ(3.49;1.16) and
pl is μ(1.27;0.60) and *pw* is μ(0.01;0.37)]$^{0.46}$
THEN y_1 is 0.40-0.21*sl*+0.35*sw*-0.39*pl*-0.04*pw* and
y2 is 0.15-0.28*sl*+0.49*sw*+0.43*pl*-0.31*pw* and
y3 is –0.25+0.25*sl*+0.04*sw*-0.85*pl*+0.20*pw*

Rule 2: IF [*sl* is μ(5.50;1.10) and *sw* is μ(2.58;0.36,0.62) and
pl is μ(4.08;1.38) and *pw* is μ(1.10;0.46,0.58)]$^{0.49}$
THEN y_1 is 0.21-0.21*sl*+0.39*sw*+0.10*pl*-0.02*pw* and
y2 is 0.63-0.20*sl*+0.49*sw*+0.10*pl*-0.86*pw* and
y3 is –0.71+0.04*sl*+0.58*sw*-0.15*pl*+0.47*pw*

Rule 3: IF [*sl* is μ(6.58;0.35,0.73,1.12,1.47,1.85) and
sw is μ(2.91;0.20,0.29,0.38,0.51,0.69) and
pl is μ(5.70;0.59,1.85) and *pw* is μ(1.56;0.65)]$^{0.54}$
THEN y_1 is 0.40-0.56*sl*-0.73*sw*+0.98*pl*+0.54*pw* and
y2 is –0.66-0.29*sl*+0.82*sw*-0.15*pl*+0.47*pw* and
y3 is 0.16+0.02*sl*+0.34*sw*-0.01*pl*-0.69*pw*

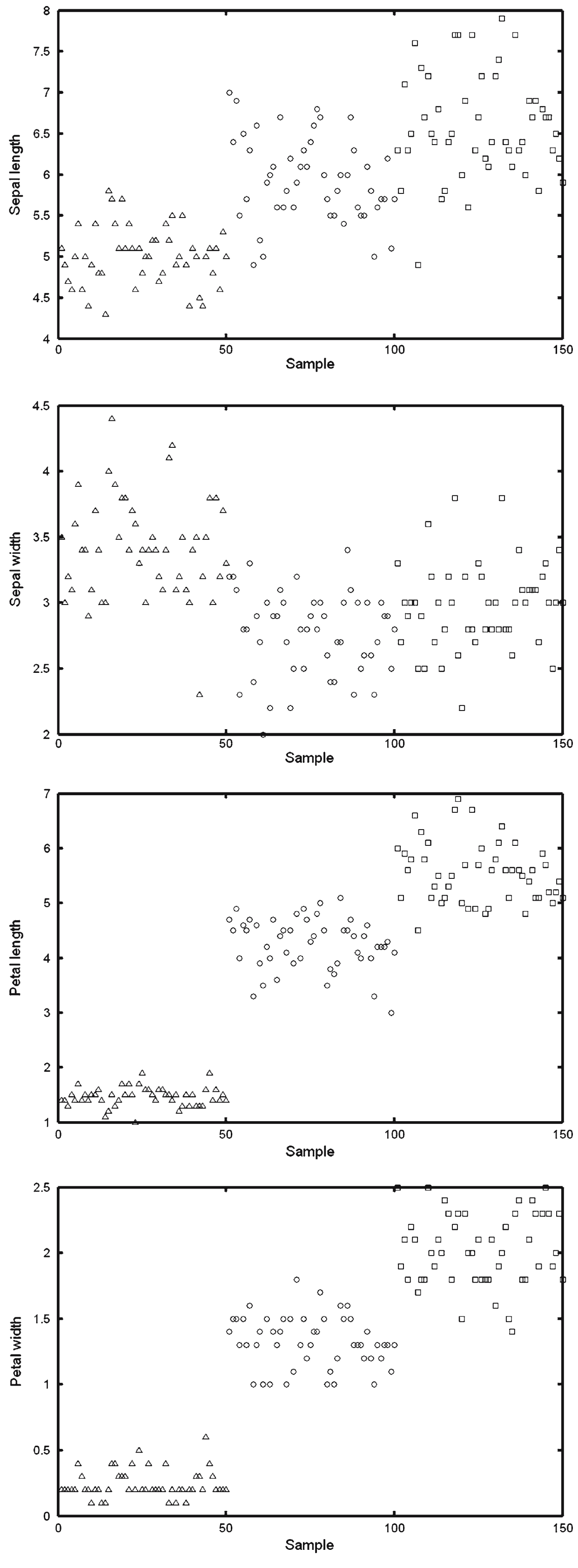


Fig. 8 Iris data: *Iris sestosa* (△), *Iris versiolor* (◯), and *Iris virginica* (□)

Table 1 The number of quantum levels for each dimension of cluster

Cluster \ No. of *ns* \ Dimension	#1	#2	#3	#4
#1	5	5	2	1
#2	1	2	1	2
#3	1	1	1	1

Table 3, we compared the learning speed (i.e., CPU time) of the QNFC model with those of the QNFC without compensatory operation, NN and RBFN. The average learning times of the QNFC, QNFC without compensatory operation, NN and RBFN were 1.0663, 1.9843, 2.127 and 4.6219 s, respectively. The average learning time was measured on a personal computer with an Intel Pentium 4 (2,500 MHz) CPU inside. Table 4 shows the comparison of the classification results of the QNFC model with other classifiers (Lee et al. 2001; Wang and George Lee 2002; Simpson 1992; Lee 1998; Wu and Chen 1999) on the Iris data. The results show that the proposed QNFC model is able to keep similar average substitution accuracy.

Example 2: Wisconsin breast cancer diagnostic data

The Wisconsin breast cancer diagnostic data set contains 699 patterns distributed into two output classes, "benign" and "malignant." Each pattern consists of nine input features: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. Four hundred fifty-eight patterns are in the benign class and the other 241 patterns are in the malignant class. Since there were 16 patterns containing missing values, we used 683 patterns to evaluate the performance of the proposed QNFC model. To compare the performance with other models, we used half of the 683 patterns as the training set and the remaining patterns as the testing set.

Experimental conditions were the same as the previous experiment. We also used half of the original data patterns as the training data (randomly selected) and the remaining patterns as the testing data. For the self-clustering algorithm (SCA), we chose the parameter $D_{thr} = 35$. Furthermore, we set the different number of quantum levels for each dimension of each cluster using quantum fuzzy entropy and tabulated them in Table 5. After the structure learning phase, two clusters were generated.

The network then entered the parameter learning phase. We set the learning rate to $\eta = 0.005$ and trained the QNFC model with different quantum levels for each dimension of

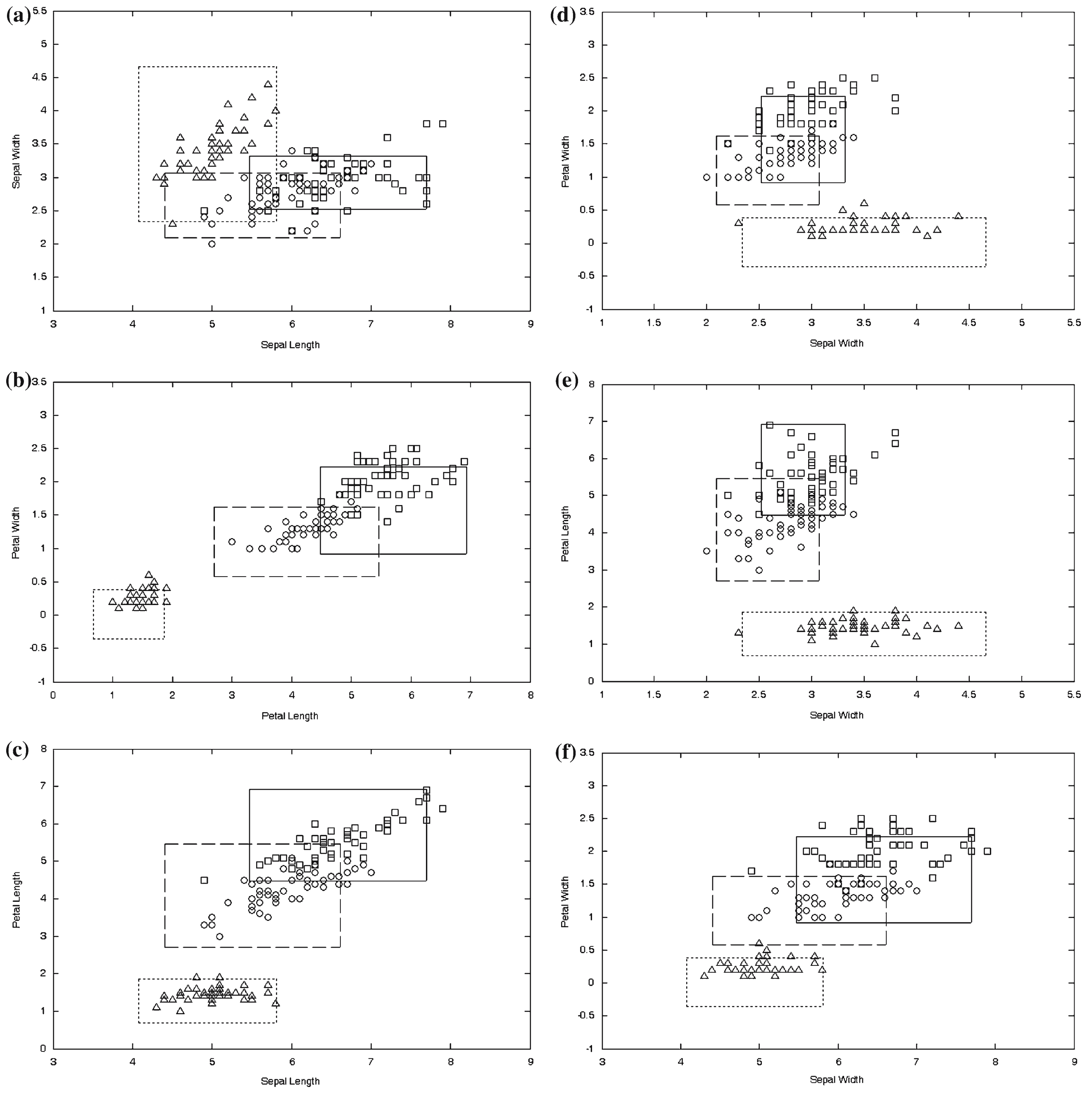


Fig. 9 The distribution of input training patterns and final assignment of three rules. **a** For the *sepal length* and *sepal width* dimensions. **b** For the *petal length* and *petal width* dimensions. **c** For the *sepal length* and *petal length* dimensions. **d** For the *sepal width* and *petal width* dimensions. **e** For the *sepal width* and *petal length* dimensions. **f** For the *sepal length* and *petal width* dimensions

each cluster. Five experiments also were used. These experiments calculated the classification accuracy and the values of the average produced on the testing set by the neural network, the RBFN with the SCA model, the QNFC model without compensatory operation, and the proposed QNFC model. During the supervised learning phase, 100 epochs of training were performed. Figure 12 shows the quantum membership functions for each input dimension. The learning curves from the proposed QNFC model, the QNFC without compensatory operation and the RBFN with the SCA model are shown in Fig. 13. The performance of the QNFC model is better than the performance of all other models.

Table 6 shows that the experiments with the QNFC model result in high accuracy, with an accuracy percentage ranging from 97.66 to 98.54%. The means of re-substitution accuracy was 97.95%. The average classification accuracy of the

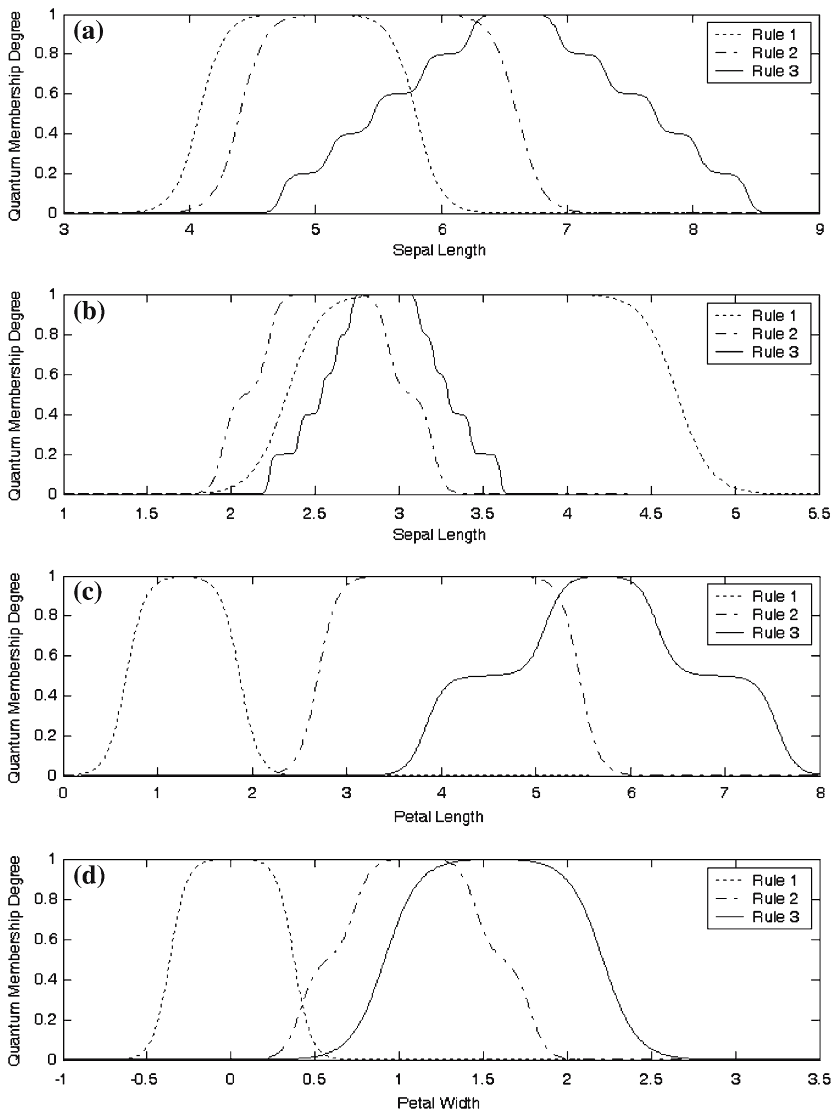


Fig. 10 Corresponding quantum membership functions for each dimension after parameter learning. **a** For the *sepal length* dimension. **b** For the *sepal width* dimension. **c** For the *petal length* dimension. **d** For the *petal width* dimension

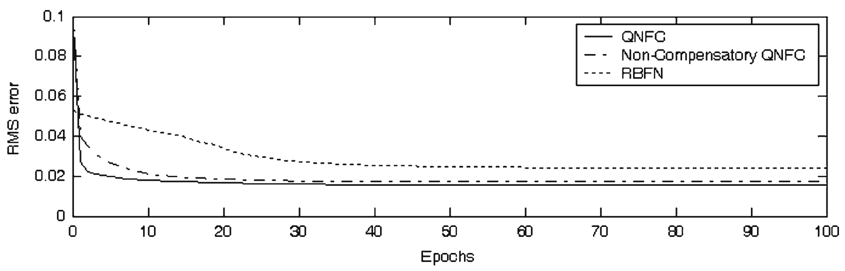


Fig. 11 Learning curves of the QNFC, the QNFC without compensatory operation, and the RBFN with the SCA model

QNFC model was better than that of other methods. Table 7 shows the CPU time of the cost of the QNFC model, the QNFC without compensatory operation, NN and RBFN. The average learning times of the QNFC, QNFC without compensatory operation, QNFC(1), QNFC(2), NN and RBFN were 4.535, 5.9781, 4.5256, 4.5351, 6.1094 and 9.5375 s, respectively. We compared the testing accuracy of our model with that of other methods (Setiono and Liu 1997; Lee et al. 2001; Nauck and Kruse 1997; Wang and George Lee 2002; Lovel and Bradley 1996). Table 8 shows the comparison between the learned QNFC models and other fuzzy, neural networks, and neuro-fuzzy classifiers. The average classification accuracy of the QNFC model is better than that of other methods.

Table 2 Classification accuracy using various methods for the Iris data

Experiment #	Model			
	Neural network	RBFN with SCA	Non-compensatory QNFC	QNFC
1	96	98.67	98.67	98.67
2	92	93.33	96	96
3	97.33	94.67	97.33	98.67
4	97.33	98.67	98.67	97.33
5	94.67	94.67	96	97.33
Average (%)	95.47	96	97.33	97.6

Table 3 The average learning time using various methods for the Iris data

Experiment #	Model			
	Neural network	RBFN with SCA	Non-compensatory QNFC	QNFC
1	4.6563	2.1406	1.9688	1.1361
2	4.6094	2.1250	2.0156	0.9837
3	4.5781	2.0781	2.0313	0.9516
4	4.6250	2.1563	1.9219	1.1203
5	4.6406	2.1350	1.9844	1.1401
Average (second)	4.6219	2.1270	1.9843	1.0663

Table 4 Average re-substitution accuracy comparison of various models for the Iris data classification problem

Methods	Average re-substitution accuracy (%)
FEBFC (Lee et al. 2001)	96.91
SANFIN (Wang and George Lee 2002)	97.33
FMMC (Simpson 1992)	97.3
FUNLVQ+GFENCE (Lee 1998)	96.3
Wu and Chen's (Wu and Chen 1999)	96.21
QNFC	97.6

Table 5 The number of quantum level for each dimension of cluster

Dimension / No. of *ns* / Cluster	#1	#2	#3	#4	#5	#6	#7	#8	#9
#1	1	1	1	1	1	2	1	1	1
#2	1	1	1	1	1	1	1	1	1

6 Conclusion

In this paper, a quantum neuro-fuzzy classifier (QNFC) was proposed for classification applications. The proposed QNFC model is a five-layer structure, which combines the compensatory-based fuzzy reasoning method with the traditional Takagi–Sugeno–Kang (TSK) fuzzy model. Compensatory operators are used to optimize fuzzy logic reasoning and to select optimal fuzzy operators. Therefore, an effective neuro-fuzzy system should be able not only to adaptively adjust fuzzy membership functions but also to dynamically optimize adaptive fuzzy operators. A self-constructing learning algorithm, which consists of the self-clustering algorithm (SCA), quantum fuzzy entropy and the backpropagation algorithm, were also proposed. Finally, simulation results were conducted to show the performance and applicability of the proposed model. The advantages of the proposed QNFC model are summarized as follows: (1) it converges quickly; (2) it is constructed automatically; (3) it has much lower rms error; and (4) it has a higher accuracy classification rate than other models.

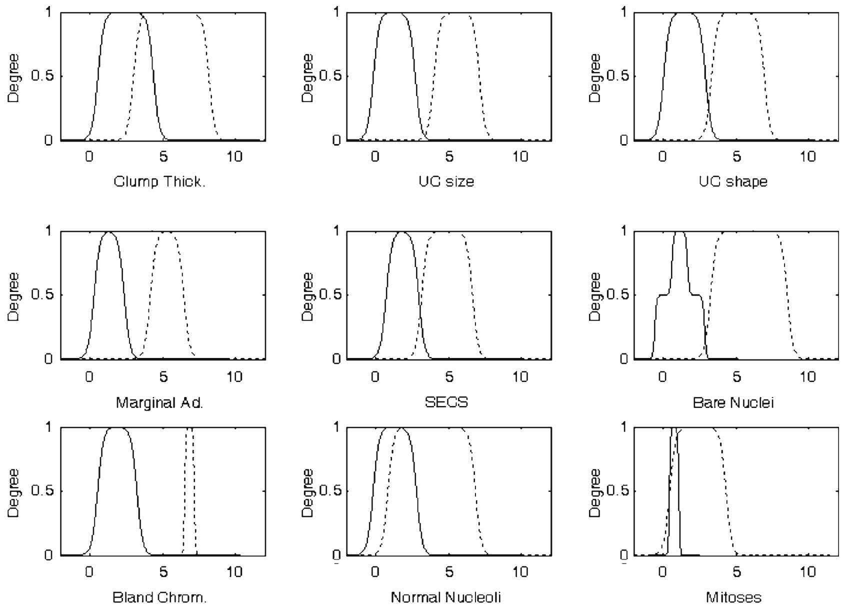


Fig. 12 Input quantum membership function for breast cancer classification (*solid line* Rule 1, *dotted line* Rule 2)

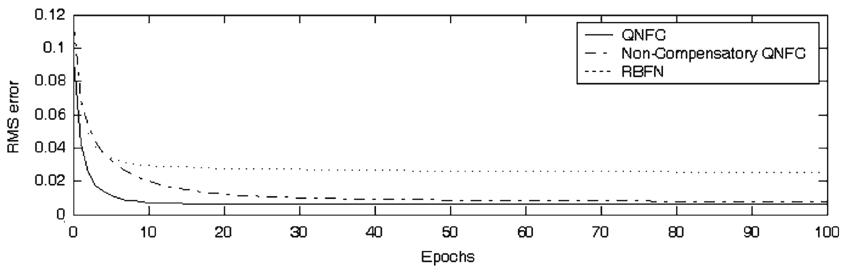


Fig. 13 Learning curves from the QNFC model, the QNFC without compensatory and the RBFN with the SCA model

Table 6 Classification accuracy for the Wisconsin breast cancer diagnostic data

Experiment #	Model					
	Neural network	RBFN with SCA	Non-compensatory QNFC	QNFC(1)[a]	QNFC(2)[b]	QNFC
1	96.49	95.32	97.66	97.66	97.66	97.66
2	97.08	95.61	98.54	98.54	98.54	98.54
3	94.44	93.86	97.37	97.66	97.37	97.66
4	97.37	94.74	97.37	97.66	97.66	97.95
5	96.49	94.74	97.66	97.66	97.66	97.95
Average (%)	96.37	94.85	97.72	97.84	97.78	97.95

[a] The QNFN model with one quantum level for all dimensions
[b] The QNFN model with two quantum levels for all dimensions

Table 7 The average learning time using various methods for the Wisconsin breast cancer diagnostic data

Experiment #	Model					
	Neural network	RBFN with SCA	Non-compensatory QNFC	QNFC(1)	QNFC(2)	QNFC
1	9.5938	6.1094	6.0156	4.6113	4.3325	4.8190
2	9.4531	6.0469	6.0781	4.7629	4.5960	4.5813
3	9.5469	6.0625	5.8750	4.3563	4.8112	4.3155
4	9.5156	6.0938	5.9688	4.4181	4.4347	4.4214
5	9.5781	6.2344	5.9531	4.4792	4.5011	4.5376
Average (second)	9.5375	6.1094	5.9781	4.5256	4.5351	4.5350

Table 8 Average accuracy comparison of various models for Wisconsin breast cancer diagnostic data

Models	Average accuracy (%)
NNFS (Setiono and Liu 1997)	94.15
FEBFC (Lee et al. 2001)	95.14
NEFCLASS (Nauck and Kruse 1997)	92.7
SANFIS (Wang and George Lee 2002)	96.3
MSC (Lovel and Bradley 1996)	94.9
QNFC	97.95

In addition to being used to solve the problems given in this paper, the proposed QNFC model was also used in our laboratory to solve practical problems on the detection of skin color and the posture classification of human body.

Acknowledgments This work was supported in part by the Ministry of Economic Affairs, Taiwan, ROC, under Grant 96-EC-17-A-02-S1-032, and the National Science Council, Taiwan, ROC, under Grant NSC 95-2221-E-009-180.

References

Duda PO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York

Fei L, Shengmei Z, Baoyu Z (2000) Quantum neural network in speech recognition. Proc IEEE Int Conf Signal Process 6:1267–1270

Halgamuge S, Glesner M (1994) Neural networks in designing fuzzy systems for real world applications. Fuzzy Sets Syst 65:1–12

Kasabov N (1996) Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. Fuzzy Sets Syst 82:135–149

Kretzschmar R, Bueler R, Karayiannis NB, Eggimann F (2000) Quantum neural networks versus conventional feedforward neural networks: an experimental study. Proc IEEE Int Conf Signal Process 1:328–337

Lee HM (1998) A neural network classifier with disjunctive fuzzy information. Neural Netw 11(6):1113–1125

Lee HM, Chen CM, Chen JM, Jou YL (2001) An efficient fuzzy classifier with feature selection based on fuzzy entropy. IEEE Trans Syst Man Cybern B 31:426–432

Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6(6):861–867

Lin CJ, Chen CH (2003) Nonlinear system control using compensatory neuro-fuzzy networks. IEICE Trans Fundam Electron Commun Comput Sci E86-A(9):2309–2316

Lin CJ, Ho WH (2003) A pseudo-Gaussian-based compensatory neural fuzzy system. IEEE International Conference on Fuzzy Systems, pp 214–220

Lin CJ, Chen CH, Lee CY (2004) A self-adaptive quantum radial basis function network for classification applications. IEEE International Joint Conference on Neural Networks, Budapest, pp 3263–3268, July 25–29

Lovel BC, Bradley AP (1996) The multiscale classifier. IEEE Trans Pattern Anal Mach Intell 18:124–137

Nauck D, Kruse R (1997) A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Sets Syst 89:277–288

Ouyang CS, Lee SJ (1999) An improved learning algorithm for rule refinement in neuro-fuzzy modeling. Third International Conference Knowledge-Based Intelligent Information Engineering Systems, pp 238–241

Paul S, Kumar S (2002) Subsethood-product fuzzy neural inference system (SuPFuNIS). IEEE Trans Neural Netw 13(3):578–599

Purushothaman G, Karayiannis NB (1997) Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks. IEEE Trans Neural Netw 8(3):679–693

Russo M (1998) FuGeNeSys—a fuzzy genetic neural system for fuzzy modeling. IEEE Trans Fuzzy Syst 6:373–388

Seker H, Evans DE, Aydin N, Yazgan E (2001) Compensatory fuzzy neural networks-based intelligent detection of abnormal neonatal cerebral doppler ultrasound waveforms. IEEE Trans Inf Technol Biomed 5(3):187–194

Setiono R, Liu H (1997) Neural-network feature selector. IEEE Trans Neural Netw 8(3):654–662

Simpson PK (1992) Fuzzy min-max neural networks—Part I: Classification. IEEE Trans Neural Netw 3:776-786

Wang JS, George Lee CS (2002) Self-adaptive neuro-fuzzy inference systems for classification applications. IEEE Trans Fuzzy Syst 10(6):790–802

Wu TP, Chen SM (1999) A new method for constructing membership functions and fuzzy rules from training examples. IEEE Trans Syst Man Cybern B 29:25–40

Zhang YQ, Kandel A (1998) Compensatory neurofuzzy systems with fast learning algorithms. IEEE Trans Neural Netw 9(1):83–105

Zimmermann HJ, Zysno P (1980) Latent connective in human decision. Fuzzy Sets Syst 4:31–51