Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2007) 2502–2516
NEUROCOMPUTING
www.elsevier.com/locate/neucom

An entropy-based quantum neuro-fuzzy inference system for classification applications

Cheng-Jian Lin[a,*], I-Fang Chung[b], Cheng-Hung Chen[c]

[a]Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taiching country, Taiwan
[b]Institute of Bioinformatics, National Yang-Ming University, Taiwan
[c]Department of Electrical and Control Engineering, National Chiao-Tung University, Taiwan

Received 13 May 2005; received in revised form 3 August 2006; accepted 7 August 2006
Available online 26 October 2006

Abstract

In this paper, an entropy-based quantum neuro-fuzzy inference system (EQNFIS) for classification applications is proposed. The EQNFIS model is a five-layer structure, which combines the traditional Takagi-Sugeno-Kang (TSK). Layer 2 of the EQNFIS 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. Simulations were conducted to show the performance and applicability of the proposed model.

Keywords: Classification; Entropy-based fuzzy model; Quantum function; Self-clustering method; Neural 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 [1]. 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 [17] 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 [9] 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 logical. When the views above are summarized, it can be said that, in contrast to pure neural or fuzzy methods, the neural fuzzy method [3,6,13,14,16,20] possesses the advantages of both neural networks and fuzzy systems. Neuro-fuzzy systems (NFS) bring the low-level

*Corresponding author.
E-mail address: cjlin@mail.cyut.edu.tw (C.-J. Lin).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.08.008

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.

Two typical types of neuro-fuzzy systems are the Mamdani-type and Takagi–Sugeno–Kang (TSK)-type neuro-fuzzy systems. For Mamdani-type neuro-fuzzy systems [11,21], the minimum fuzzy implication is used in fuzzy reasoning. For TSK-type neuro-fuzzy systems [4,5,19], the antecedent is defined in the same way as the Mamdani-type, while the consequent is a linear function of the input variables. Many researchers [4,5] have shown that using a TSK-type neuro-fuzzy system achieves superior performance in network size and learning accuracy than using Mamdani-type neuro-fuzzy systems.

Recently, quantum neural networks (QNNs) used to limit conventional neural networks (NNs) were developed [2,7,15]. Conventional NNs and QNNs satisfy the requirements outlined in [10] 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, an entropy-based quantum neuro-fuzzy inference system (EQNFIS) is proposed. The EQNFIS model is a five-layer structure, which combines the traditional TSK. Layer 2 of the EQNFIS 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.

A self-constructing learning algorithm for the EQNFIS 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 EQNFIS model using two classification problems.

This paper is organized as follows. Section 2 describes the quantum membership function and the structure of the EQNFIS model. Section 3 describes the learning algorithm of the EQNFIS model. The self-clustering algorithm, quantum fuzzy entropy, and backpropagation algorithm are presented in this section. In Section 4, the EQNFIS 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 structure of the EQNFIS

The fuzzy if-then rule shown below is used by the EQNFIS:

$$Rj: \text{IF } x_1 \text{ is } Q_{1j} \text{ and } \ldots\ldots \text{and } x_n \text{ is } Q_{nj}$$
$$\text{THEN } y \text{ is } a_{0j} + \sum_{i=1}^{n} a_{ij} x_i, \tag{1}$$

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 $\mu_{Q_{ij}}$; a_{0j} and a_{ij} are the parameters of consequent part; n is the number of input dimensions; R_j is jth fuzzy rule.

The membership function of the precondition part discussed in this paper is different from the typical Gaussian membership function. We adopt the quantum membership function to approximate desired results. Therefore, the response of the jth quantum membership function for the ith feature vector can be written as

$$Q_{ij} = \frac{1}{ns_{ij}} \sum_{r=1}^{ns_{ij}} \left[\left(\frac{1}{1+\exp(-\beta(x_i - m_{ij} + |\theta_{ij}^r|))} \right) U(x_i; -\infty, m_{ij}) + \left(\frac{\exp(-\beta(x_i - m_{ij} - |\theta_{ij}^r|))}{1+\exp(-\beta(x_i - m_{ij} - |\theta_{ij}^r|))} \right) U(x_i; m_{ij}, \infty) \right], \tag{2}$$

where $U(x_i; a, b) = \begin{cases} 1 & \text{if } a \leqslant x_i < b \\ 0 & \text{otherwise} \end{cases}$, β is the slope factor, θ_{ij}^r 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:

$$Rj: \text{IF } x_1 \text{ is } \mu(m_{1j}; \theta_{1j}^{r_{1j}}) \text{ and } \ldots \text{and } x_i \text{ is } \mu(m_{ij}; \theta_{ij}^{r_{ij}}) \text{ and} \ldots \text{ and } xn \text{ is } \mu(m_{nj}; \theta_{nj}^{r_{nj}})$$
$$\text{THEN } y \text{ is } a_{0j} + a_{1j}x_1 + \ldots + a_{ij}x_i + \ldots + a_{nj}x_n. \tag{3}$$

Fig. 1 shows the response of a three-level quantum membership function.

The structure of the EQNFIS, 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 EQNFIS 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.

$$u_i^{(1)} = x_i. \tag{4}$$

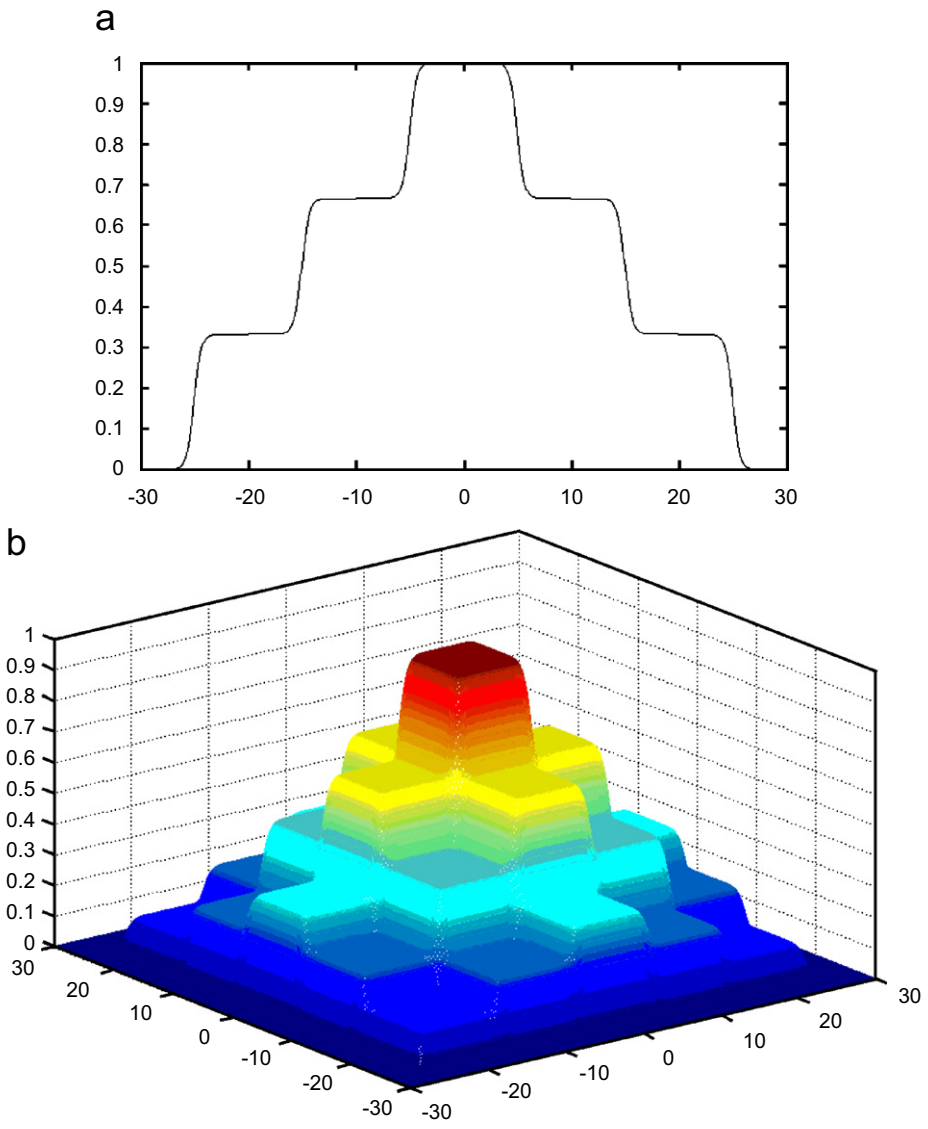

Fig. 1. Quantum membership function shown in (a) one-dimension (b) two-dimensions.

Layer 2 (Membership Function Node)*:* Nodes in this layer correspond to one linguistic label of the input variables in layer 1 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, the operation performed in layer 2 is

$$u_{ij}^{(2)} = \frac{1}{ns_{ij}} \sum_{r=1}^{ns_{ij}} \left[\left(\frac{1}{1+\exp(-\beta(u_i^{(1)} - m_{ij} + |\theta_{ij}^r|))} \right) U(u_i^{(1)}; -\infty, m_{ij}) + \left(\frac{\exp(-\beta(u_i^{(1)} - m_{ij} - |\theta_{ij}^r|))}{1+\exp(-\beta(u_i^{(1)} - m_{ij} - |\theta_{ij}^r|))} \right) U(u_i^{(1)}; m_{ij}, \infty) \right], \tag{5}$$

where $U(x_i; a, b) = \begin{cases} 1 & \text{if } a \leqslant x_i < b \\ 0 & \text{otherwise} \end{cases}$, β is the slope factor, θ_{ij}^r 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 (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 the nodes of a set in layer 2. Here, we use the product operator mentioned above to perform IF-condition matching of fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \left(\prod_i u_{ij}^{(2)} \right), \tag{6}$$

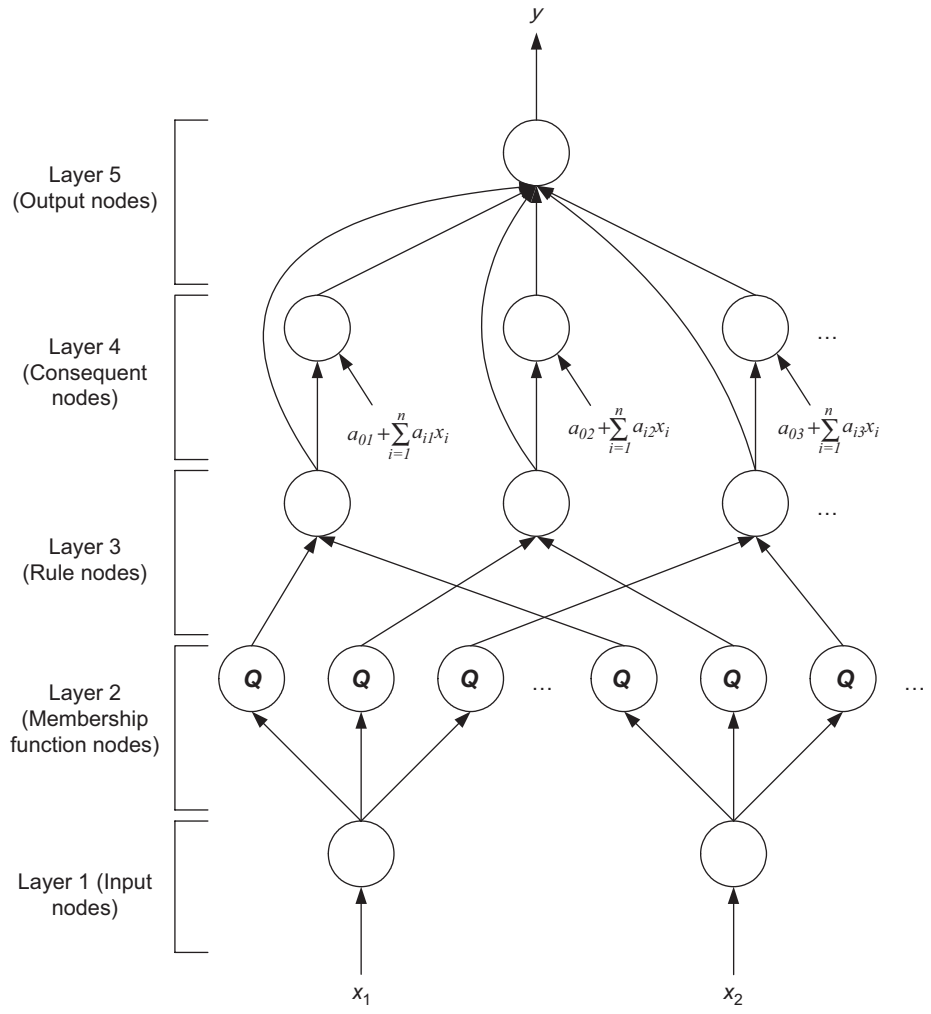

Fig. 2. Structure of the proposed EQNFIS.

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule.

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

$$u_j^{(4)} = u_j^{(3)}\left(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i\right), \tag{7}$$

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

$$y = u^{(5)} = \frac{\sum_{j=1}^{p} u_j^{(4)}}{\sum_{j=1}^{p} u_j^{(3)}} = \frac{\sum_{j=1}^{p} u_j^{(3)}\left(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i\right)}{\sum_{j=1}^{p} u_j^{(3)}}, \tag{8}$$

where p is the number of the fuzzy rule.

3. A learning algorithm for the EQNFIS model

In this section, we present a learning algorithm for the proposed EQNFIS model. The following two schemes are part of this learning algorithm. 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 decide the number of quantum levels that reflect the actual distribution of classification patterns. Second, a supervised learning scheme is used 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.

3.1. Structure learning

The first step in structure learning is to determine the number of rules using the SCA from the training data, as well as to determine the number of fuzzy sets in the universal of discourse for each input variable, since one

cluster in the input space corresponds to one potential fuzzy logic rule, with m_{ij} and θ_{ij}^{r} representing the center and the quantum interval, respectively. Simultaneously, we employ quantum fuzzy entropy to determine the appropriate number of quantum levels. After the SCA, the quantum intervals and the number of quantum levels are determined. It is then easy to decide on the quantum membership function.

3.1.1. The self-clustering algorithm

Layer 2 of the EQNFIS 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 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. The notations in the SCA are described as follows:

P_i	the ith input sample
C	the cluster center
C_j	the jth cluster center
R	the number of cluster
C_{m}	the current sample P_i belongs to the cluster with the minimum distance
Dc	the diagonal distance of cluster C
Wc	the boundary width of cluster C
D_{thr}	the threshold value of the distance
$Dist_{ij}$	the distance between the current sample P_i and the jth cluster center
$Dist_{\mathrm{im}}$	the distance between the current sample P_i and the cluster center C_{m} with the minimum distance
x, y	x- and y-dimension in the diagram

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 Wc, 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. Generally, D_{thr} is set from 0.5 to 1 time the summation of the samples variance in this study.

In this paper, we use two-dimensional feature spaces as an example to explain the proposed clustering algorithm.

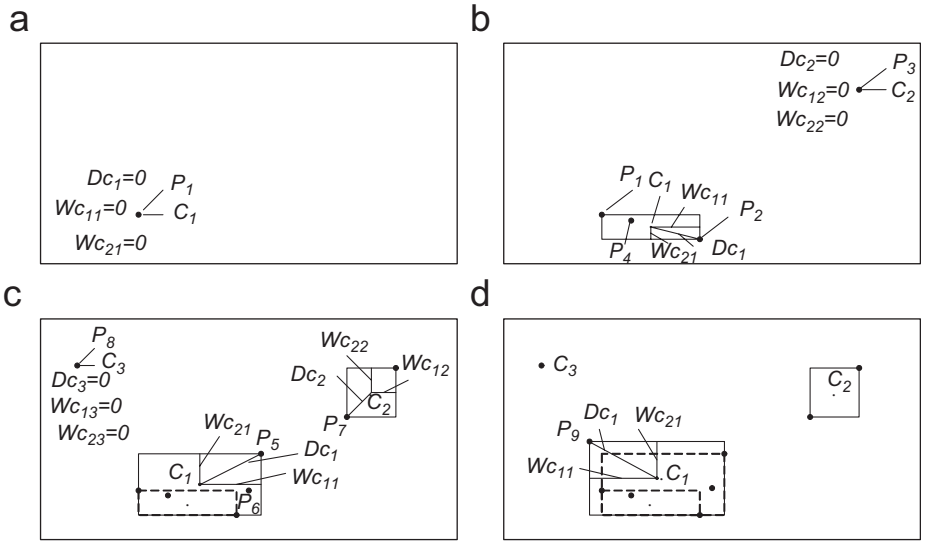

Fig. 3. A brief clustering process using the SCA with samples P_1 to P_9 in 2-D space. (a) The sample P_1 causes the SCA to create a new cluster center C_1. (b) P_2: update cluster center C_1, P_3: create a new cluster center C_2, P_4: do nothing. (c) P_5: update cluster C_1, P_6: do nothing, P_7: update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

Fig. 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 Wc_{11} and Wc_{21} 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 R already created cluster centers C_j, $Dist_{ij} = ||P_i - C_j||$, $j = 1,2,\ldots,R$, 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, $j = 1,2,\ldots,R$, it means that the current sample P_i belongs to a cluster C_m with the minimum distance

$$Dist_{\text{im}} = ||P_i - C_m|| = \min(||P_i - C_j||), j = 1, 2, \ldots, R. \tag{9}$$

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 R existing cluster centers by calculating the values $S_{ij} = Wc_{ij} + Dc_j$, $j = 1,2,\ldots,R$, and then choosing the cluster center C_{m} with the minimum value S_{im}:

$$S_{\text{im}} = Wc_{\text{im}} + Dc_m = \min(S_{ij}), j = 1, 2, \ldots, R. \tag{10}$$

In Eq. (9), 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. (10). But the special situation shows that the distances between a given point P_{10} and both cluster centers $Dist_{10,1}$ and $Dist_{10,2}$ are the same as shown in Fig. 4. In the aforementioned technique, the cluster C_2, which has small dimension distances Dc_2, will be selected to expand according to Eq. (10). However, this causes a problem in that the cluster numbers increase quickly. To avoid this problem, we make a judgment, as follows:

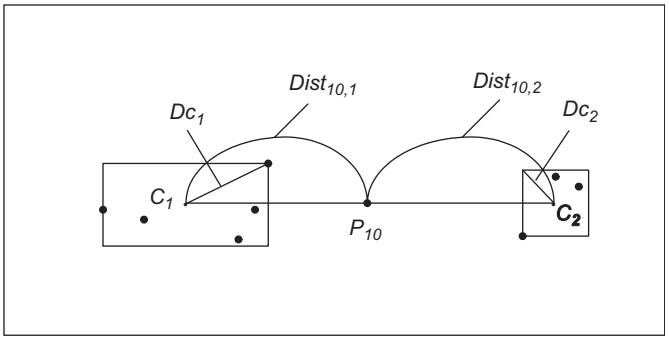

Fig. 4. The special case of SCA.

If (the distance and $Dist_{10,1}$ is equal to the distance and $Dist_{10,2}$)
and ($Dc_1 > Dc_2$)
Then $Dc_{\text{m}} = Dc_1$.

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 Dc_1.

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 $Wc_{1\text{m}}$, $Wc_{2\text{m}}$. The parameters are updated by the following equation:

$$Wc_{1\text{m}}^{\text{new}} = \frac{(||C_{\text{m}\_x} - P_{i\_x}|| + Wc_{1\text{m}})}{2}, \tag{11}$$

$$Wc_{2\text{m}}^{\text{new}} = \frac{(||C_{\text{m}\_y} - P_{i\_y}|| + Wc_{2\text{m}})}{2}, \tag{12}$$

$$C_{\text{m}\_x}^{\text{new}} = ||P_{i\_x} - D_{1\text{m}}^{\text{new}}||, \tag{13}$$

$$C_{\text{m}\_y}^{\text{new}} = ||P_{i\_y} - D_{2\text{m}}^{\text{new}}||, \tag{14}$$

$$Dc_{\text{m}}^{\text{new}} = S_{\text{im}}/2, \tag{15}$$

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 number of rules, the center and the quantum interval of the quantum membership function are defined by the following equation:

$$m_{ij} = C_j, j = 1, 2, \ldots, R, \tag{16}$$

$$\theta_{ij}^{r} = \frac{1}{((ns_{ij} + 1)/2)} rD_j,$$
$$r = 1, 2, \ldots, ns_{ij},\ j = 1, 2, \ldots, R \tag{17}$$

$$R = \text{the number of clusters} \tag{18}$$

3.1.2. Quantum fuzzy fntropy

After that, the center and the quantum interval of the quantum membership function are determined. The number of quantum levels in each dimension has

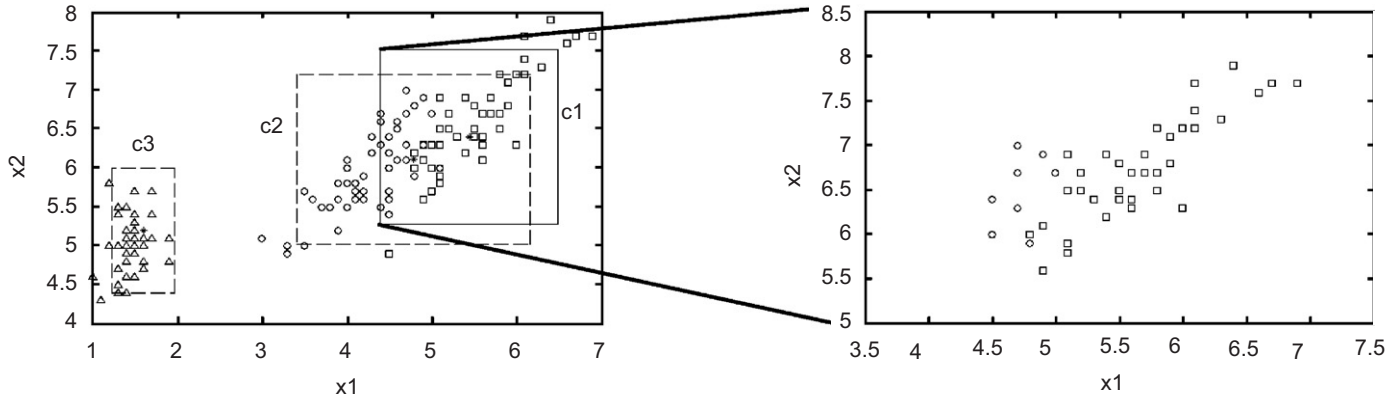

Fig. 5. The pattern distribution with two dimensions and three classes of the cluster C_1.

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 proposed criterion is based on quantum fuzzy entropy, since it has the ability to reflect the actual distribution of pattern space. Fig. 5 briefly shows distribution of the pattern space for a cluster after the SCA clustering process in two-input space, and that describes our proposed quantum fuzzy entropy of the quantum interval for each dimension of the cluster. 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 2. We compute the quantum fuzzy entropy for all clusters in each dimension to obtain the distribution information of patterns projected in this dimension. Quantum fuzzy entropy is defined as follows:

(1) Let $X = \{x_1, x_2, \ldots, x_{nc}\}$ be a classification set with elements x_i distributed in a pattern space, where $i = 1; 2; \ldots; nc$.
(2) Let $\tilde{Q}$ be a quantum fuzzy set defined in the quantum interval 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 $CL_1; CL_2; \ldots; CL_p$ represent p classes into which the n elements are divided.
(4) Let $T_{CL_j}(x_{nc})$ 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 interval, where $j = 1; 2; \ldots; p$, is defined as

$$SD_j = \frac{\sum_{x \in T_{CL_j}(x_{nc})} \mu_{\tilde{Q}}(x)}{\sum_{x \in X} \mu_{\tilde{Q}}(x)}. \tag{19}$$

(6) The quantum fuzzy entropy $QFE_{CL_j}(\tilde{Q})$ of the elements of class j in the quantum interval is defined as

$$QFE_{CL_j}(\tilde{Q}) = -SD_j \log SD_j. \tag{20}$$

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

$$QFE(\tilde{Q}) = \sum_{j=1}^{p} QFE_{CL_j}(\tilde{Q}). \tag{21}$$

(8) In this step, we can compute the quantum fuzzy entropy for the quantum levels $ns = 1$ and $ns = 2$, as shown in Fig. 6.

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, go to *Step 6*.

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.

3.1.3. The parameter learning

After the network structure is determined by the SCA, the network then enters the parameter learning phase to

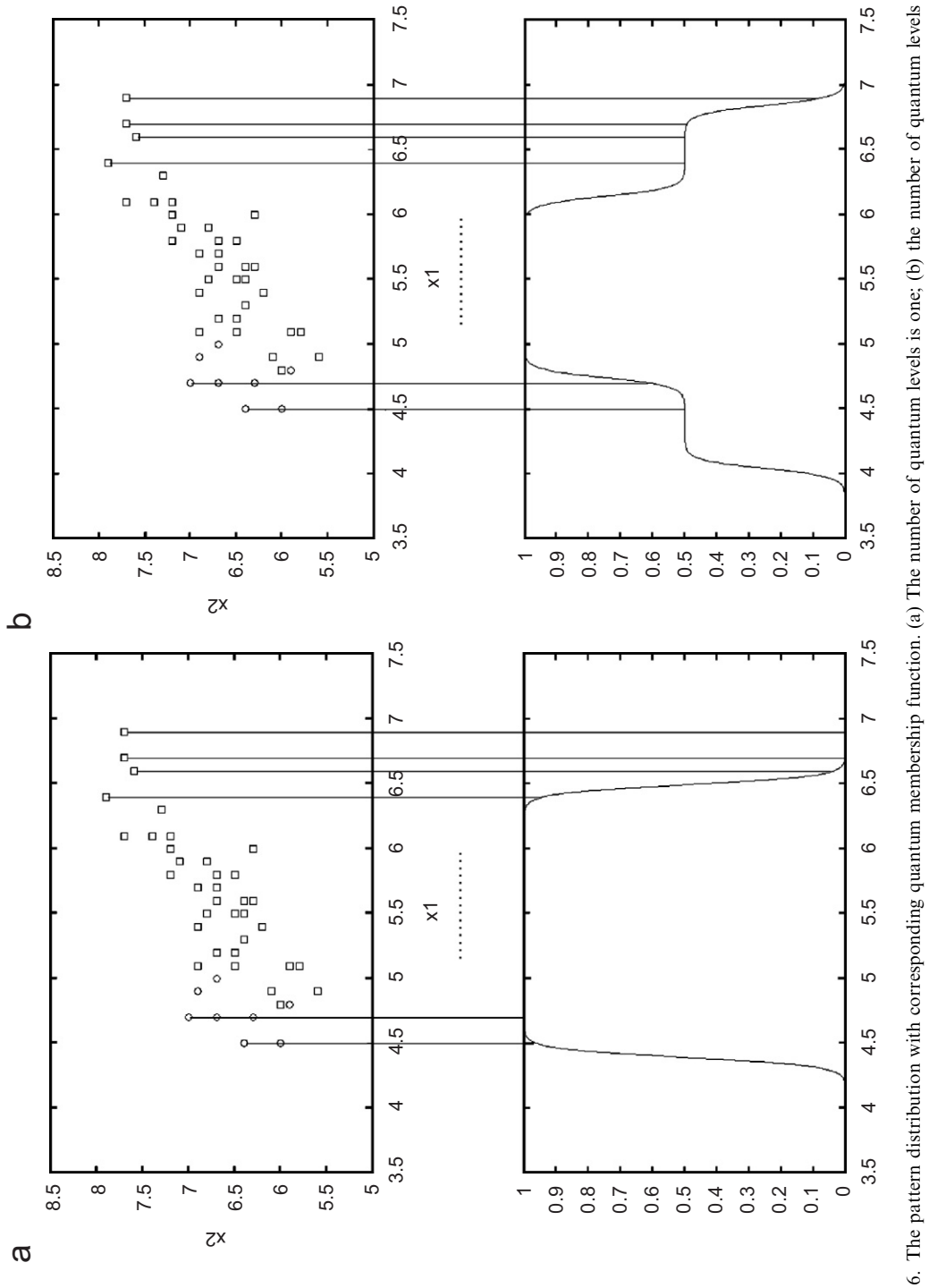

Fig. 6. The pattern distribution with corresponding quantum membership function. (a) The number of quantum levels is one; (b) the number of quantum levels is two.

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 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

$$E = \frac{1}{2}[y - y^d]^2, \tag{22}$$

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

$$\delta_e = -\frac{\partial E}{\partial y} = y^d - y. \tag{23}$$

The parameter of consequent part is updated by the amount

$$\Delta a_{0j} = -\frac{\partial E}{\partial a_{0j}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial u_j^{(4)}}\right]\left[\frac{\partial u_j^{(4)}}{\partial a_{0j}}\right] = \frac{\delta_e u_j^{(3)}}{\sum_{j=1}^{p} u_j^{(3)}} \tag{24}$$

and

$$\Delta a_{ij} = -\frac{\partial E}{\partial a_{ij}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial u_j^{(4)}}\right]\left[\frac{\partial u_j^{(4)}}{\partial a_{ij}}\right] = \frac{\delta_e u_j^{(3)} x_i}{\sum_{j=1}^{p} u_j^{(3)}}. \tag{25}$$

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

$$a_{0j}(t+1) = a_{0j}(t) + \eta_a \Delta a_{0j}, \tag{26}$$

$$a_{ij}(t+1) = a_{ij}(t) + \eta_a \Delta a_{ij}, \tag{27}$$

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 the quantum function neurons of the hidden layer to update their centers and quantum intervals. According to the chain rule, the updated center is as follows:

$$\Delta m_{ij} = -\frac{\partial E}{\partial m_{ij}} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial m_{ij}}\right] = \delta_e \cdot \left[\frac{(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i) \cdot \sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}(u_j^{(3)} \cdot (a_{0j} + \sum_{i=1}^{n} a_{ij}x_i))}{(\sum_{j=1}^{p} u_j^{(3)})^2}\right] \prod_{\substack{j=1 \\ i \neq j}}^{p} Q_{ij}$$
$$\times \frac{1}{ns_{ij}} \sum_{r=1}^{ns_{ij}} \left[-\frac{\beta \cdot (\exp(-\beta \cdot (x_i - m_{ij} + |\theta_{ij}^r|)))}{(1 + \exp(-\beta \cdot (x_i - m_{ij} + |\theta_{ij}^r|)))^2} \cup (x_i; -\infty, m_{ij}) + \frac{\beta \cdot (\exp(-\beta \cdot (x_i - m_{ij} + |\theta_{ij}^r|)))}{(1 + \exp(-\beta \cdot (x_i - m_{ij} + |\theta_{ij}^r|)))^2} \cup (x_i; m_{ij}, \infty)\right]. \tag{28}$$

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

$$\Delta \theta_{ij}^r = -\frac{\partial E}{\partial \theta_{ij}^r} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial \theta_{ij}^r}\right] = \delta_e \cdot \left[\frac{(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i) \cdot \sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}(u_j^{(3)} \cdot (a_{0j} + \sum_{i=1}^{n} a_{ij}x_i))}{(\sum_{j=1}^{p} u_j^{(3)})^2}\right] \prod_{\substack{j=1 \\ i \neq j}}^{p} Q_{ij}$$
$$\times \frac{1}{ns_{ij}} \cdot \left[\frac{\beta \cdot (\exp(-\beta \cdot (x_i - m_{ij} + \theta_{ij}^r)))}{(1 + \exp(-\beta \cdot (x_i - m_{ij} + \theta_{ij}^r)))^2} \cup (x_i; -\infty, m_{ij}) - \frac{\beta \cdot (\exp(-\beta \cdot (x_i - m_{ij} + \theta_{ij}^r)))}{(1 + \exp(-\beta \cdot (x_i - m_{ij} + \theta_{ij}^r)))^2} \cup (x_i; m_{ij}, \infty)\right] \tag{29}$$

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

$$\Delta \theta_{ij}^r = -\frac{\partial E}{\partial \theta_{ij}^r} = \left[-\frac{\partial E}{\partial u^{(5)}}\right]\left[\frac{\partial u^{(5)}}{\partial \theta_{ij}^r}\right] = \delta_e \cdot \left[\frac{(a_{0j} + \sum_{i=1}^{n} a_{ij}x_i) \cdot \sum_{j=1}^{p} u_j^{(3)} - \sum_{j=1}^{p}(u_j^{(3)} \cdot (a_{0j} + \sum_{i=1}^{n} a_{ij}x_i))}{(\sum_{j=1}^{p} (u_j^{(3)})^2}\right] \prod_{\substack{j=1 \\ i \neq j}}^{p} Q_{ij}$$
$$\times \frac{1}{ns_{ij}} \cdot \left[-\frac{\beta \cdot (\exp(-\beta \cdot (x_i - m_{ij} - \theta_{ij}^r)))}{(1 + \exp(-\beta \cdot (x_i - m_{ij} - \theta_{ij}^r)))^2} \cup (x_i; -\infty, m_{ij}) + \frac{\beta \cdot (\exp(-\beta \cdot (x_i - m_{ij} - \theta_{ij}^r)))}{(1 + \exp(-\beta \cdot (x_i - m_{ij} - \theta_{ij}^r)))^2} \cup (x_i; m_{ij}, \infty)\right]. \tag{30}$$

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

$$m_{ij}(t+1) = m_{ij}(t) + \eta_m \Delta m_{ij}, \tag{31}$$

$$\theta_{ij}^r(t+1) = \theta_{ij}^r(t) + \eta_\theta \Delta \theta_{ij}^r, \tag{32}$$

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

4. Illustrative examples

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

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

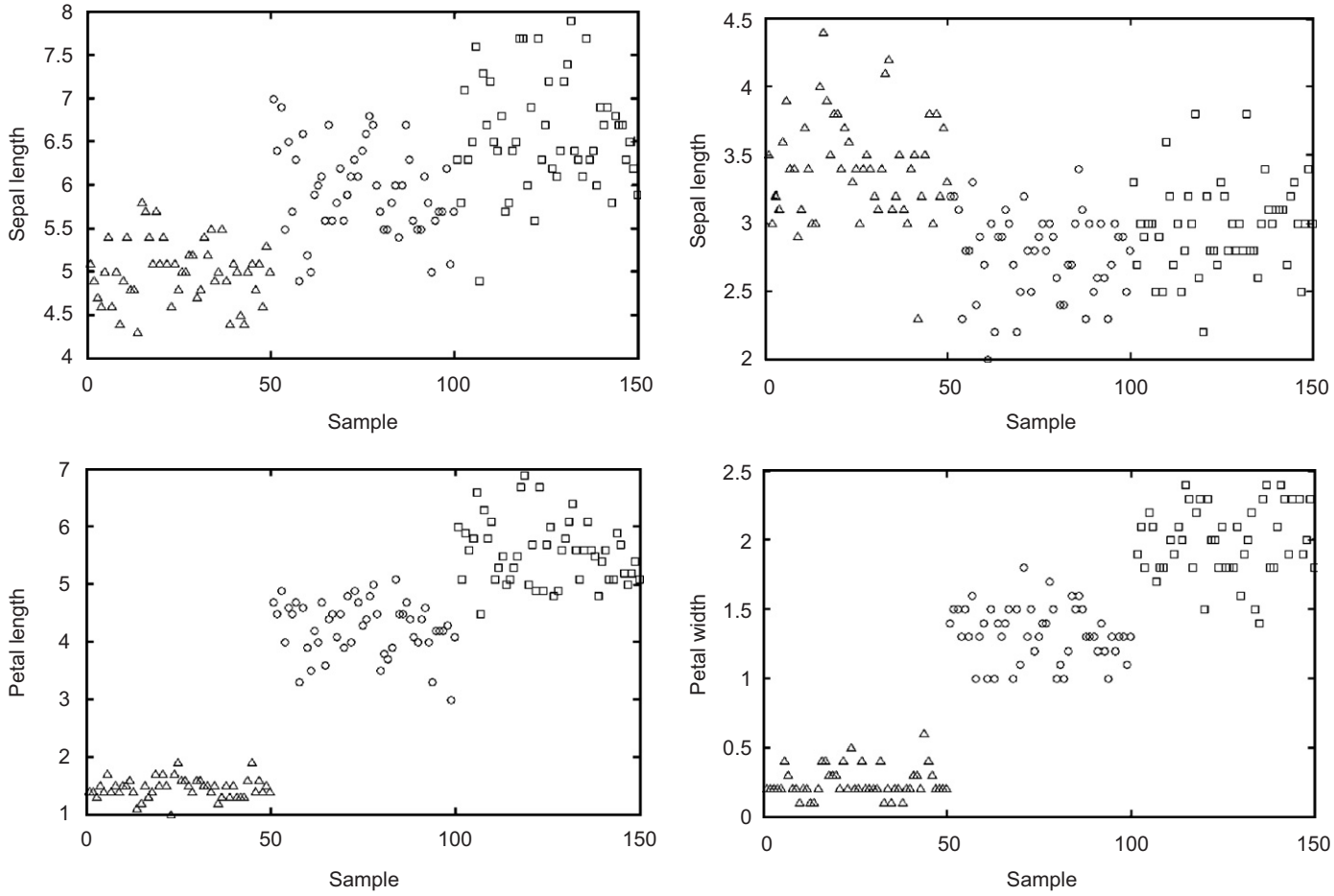

Fig. 7. Iris data: *Iris sestosa* (△), *Iris versiolor* (○), and *Iris virginica* (□).

4.1. 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*)—of 150 specimens of the iris plant. Three species of the iris were used: *Iris sestosa*, *Iris versiolor*, and *Iris virginica*. Fifth instances of each species were included. The measurements are shown in Fig. 7.

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 SCA, we chose the parameter $D_{\text{thr}} = 4.5$. 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 EQNFIS model with different quantum levels for each dimension of each cluster. After 100 training steps, the final root-mean-square (RMS) error was 0.0138. Three fuzzy logic rules were generated. The three designed fuzzy rules were:

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

No. of *ns* cluster	Dimension			
	#1	#2	#3	#4
#1				
#2				
#3	1	1	1	1

Rule 1: IF *sl* is μ(6.38;0.45,0.77,1.12,1.50,1.87) and
sw is μ(2.64;0.17,0.30,0.45,0.62,0.79) and
pl is μ(5.77;0.91,1.46) and *pw* is μ(1.77;0.44)
THEN y_1 is $-4.9 + 0.27sl\text{-}0.36sw\text{-}0.03pl + 0.46pw$ and
y2 is $-0.20\text{-}0.36sl + 0.32sw + 0.57pl\text{-}0.78pw$ and
y3 is $0.91\text{-}0.20sl\text{-}0.33sw + 0.43pl\text{-}0.51pw$

Rule 2: IF *sl* is μ(5.52;1.09) and *sw* is μ(2.89;0.26,0.40) and
pl is μ(4.09;1.37) and *pw* is μ(1.29;0.29,0.55)
THEN y_1 is $-0.49\text{-}0.01sl + 0.45sw\text{-}0.04pl\text{-}0.15pw$ and
y2 is $-0.27 + 0.41sl\text{-}0.67sw + 0.05pl\text{-}0.33pw$ and

$y3$ is $-0.65+0.20sl+0.27sw+0.12pl$-$0.67pw$
Rule 3: IF sl is $\mu(4.90;0.90)$ and sw is $\mu(3.30;1.36)$ and pl is $\mu(1.29;0.58)$ and pw is $\mu(0.18;-0.53)$
THEN y_1 is $-0.03+0.28sl$-$0.22sw$-$0.30pl$-$0.79pw$ and $y2$ is $0.07+0.79sl$-$0.57sw$-$0.68pl$-$0.72pw$ and $y3$ is $0.63+0.62sl$-$0.71sw$-$0.89pl+0.09pw$

Fig. 8(a)–(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. We compared the testing accuracy of our model with that of other methods—the traditional

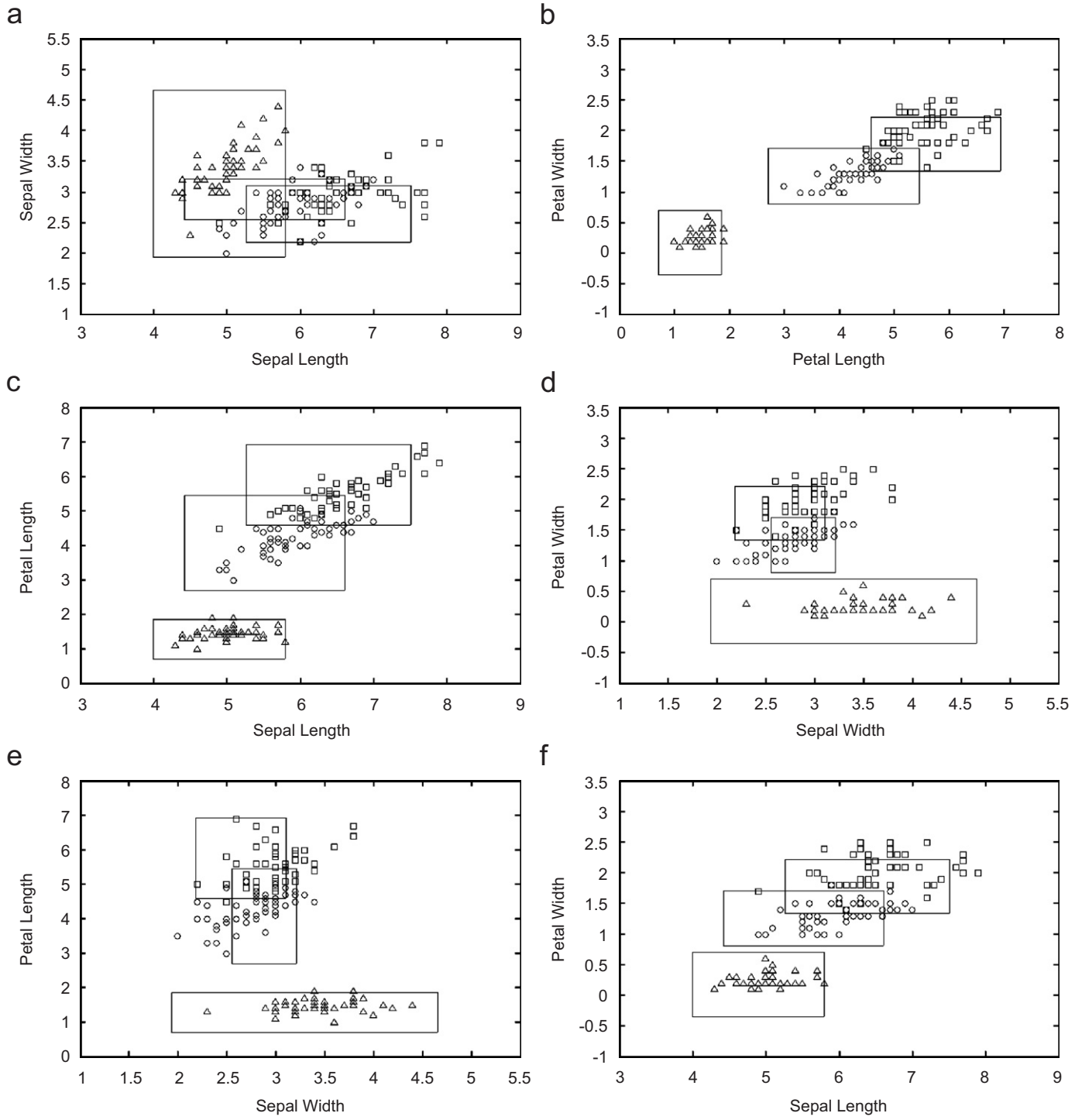

Fig. 8. The distribution of input training patterns and final assignment of three rules. (a) For the *Sepal Length* and *Sepal Width* dimensions. (b) For the *Petal Length* and *Petal Width* dimensions. (c) For the *Sepal Length* and *Petal Length* dimensions. (d) For the *Sepal Width* and *Petal Width* dimensions. (e) For the *Sepal Width* and *Petal Length* dimensions. (f) For the *Sepal Length* and *Petal Width* dimensions.

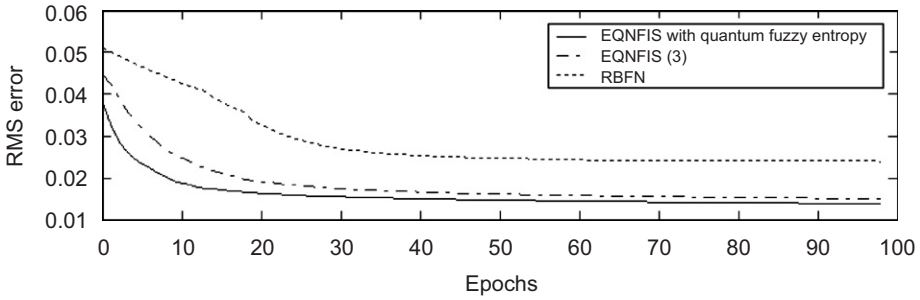

Fig. 9. Learning curves of the EQNFIS with quantum fuzzy entropy, the EQNFIS with the three quantum levels for each dimension of each cluster, and the RBFN with the SCA model.

Table 2
Classification accuracy using various methods for the Iris data

Experiment #	Model					
	Neural network	RBFN with SCA	EQNFIS (2)	EQNFIS (3)	EQNFIS (5)	EQNFIS with quantum fuzzy entropy
1						
2	92	93.33	94.67	96	94.67	96
3	97.33	94.67	94.67	96	96	97.33
4	97.33	98.67	97.33	97.33	97.33	98.67
5	94.67	94.67	94.67	96	94.67	96
Average (%)	95.47	96	96	96.8	96.27	**97.33**

multiplayer neural network (NN) with 12 hidden nodes and 84 parameters, the standard radial basis function network (RBFN) with the SCA including 8 hidden nodes and 88 parameters, and the EQNFIS with 3 fuzzy rules and 80 parameters—using the same quantum levels ($ns = 2$, 3 and 5) for each dimension of each cluster. Five experiments were used. These experiments calculated the classification accuracy and the values of the average produced on the testing set using the traditional multiplayer NN, the RBFN with the SCA, the EQNFIS model, using 2, 3, and 5 quantum levels, and the proposed EQNFIS with quantum fuzzy entropy.

During the learning phase, 100 epochs of training were performed. The learning curves from the proposed EQNFIS model with quantum fuzzy entropy, the EQNFIS with the three quantum levels for each dimension of each cluster, and the RBFN with the SCA model are shown in Fig. 9. The figure reveals a smaller rms error and a faster convergence for the EQNFIS model compared to the RBFN model. In this example, five experiments were used. These experiments have different orders of training samples. Table 2 shows that the experiments with the EQNFIS model for five different orders of data samples, having an accuracy percentage ranging from 96% to 98.67%. The means of re-substitution accuracy was 97.33%. The average classification accuracy of the EQNFIS model with quantum fuzzy entropy was better than that of other methods. In Table 3, we compared the learning speed (i.e., CPU time) of the EQNFIS model with those of the NN and RBFN. The average learning times of the EQNFIS, NN and RBFN were 1.9843, 2.127 and 4.6219 s, respectively. The average learning time was measured on a personal computer with an Intel Pentium 4 (2500 MHz) CPU inside. Table 4 shows the comparison

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

Experiment #	Model		
	Neural network	RBFN with SCA	EQNFIS with quantum fuzzy entropy
1	4.6563	2.1406	1.9688
2	4.6094	2.1250	2.0156
3	4.5781	2.0781	2.0313
4	4.6250	2.1563	1.9219
5	4.6406	2.1350	1.9844
Average (second)	4.6219	2.1270	**1.9843**

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

Methods	Average re-substitution accuracy (%)
FEBFC [9]	96.91
SANFIN [20]	97.33
FMMC [18]	97.3
FUNLVQ+GFENCE [8]	96.3
Wu and Chen's [22]	96.21
EQNFIS	**97.33**

of the classification results of the EQNFIS model with other classifiers [8,9,18,20,22] on the *Iris* data. The results show that the average classification accuracy of the EQNFIS model is better than other methods.

4.2. 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. A total of 458 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 EQNFIS 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.

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

No. of *ns* cluster	Dimension								
	#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

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 SCA, we chose the parameter $D_{\text{thr}} = 35$. Furthermore, we determined the different number of quantum levels for each dimension of each cluster using quantum fuzzy entropy and tabulated in them 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 $\eta = 0.05$ and trained the EQNFIS model with different quantum levels for each dimension of 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 with 7 hidden nodes and 77 parameters, the RBFN with the SCA including 4 hidden nodes and 80 parameters, the EQNFIS model with 2 fuzzy rules and 77 parameters using 1 and 2 quantum levels, and the proposed EQNFIS with quantum fuzzy entropy. During the supervised learning phase, 100 epochs of training were performed. Fig. 10 shows the learning curves from the proposed EQNFIS model with quantum fuzzy entropy, the EQNFIS with the two quantum levels for each dimension of each cluster, and the RBFN with the SCA model. Our model can obtain a smaller rms error and converge more quickly.

Table 6 shows that the experiments with the EQNFIS model with quantum fuzzy entropy result in high accuracy, with an accuracy percentage ranging from 97.37% to

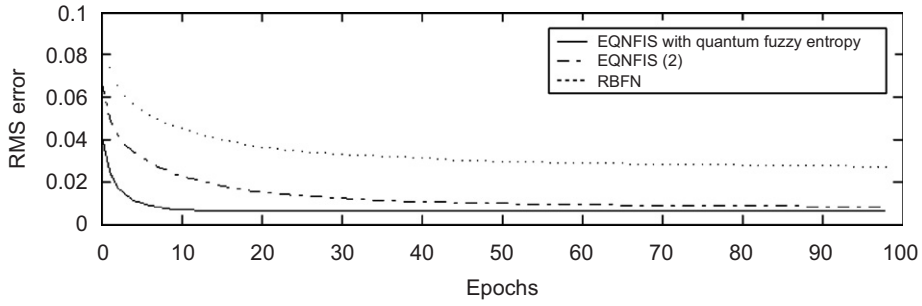

Fig. 10. Learning curves from the EQNFIS with quantum fuzzy entropy, the EQNFIS with the two quantum levels for each dimension of each cluster, and the RBFN with the SCA model.

Table 6
Classification accuracy for the Wisconsin breast cancer diagnostic data

Experiment #	Model				
	Neural network	RBFN with SCA	EQNFIS(1)	EQNFIS(2)	EQNFIS with quantum fuzzy entropy
1	96.49	95.32	97.37	97.37	97.66
2	97.08	95.61	97.66	98.54	98.54
3	94.44	93.86	97.37	97.37	97.37
4	97.37	94.74	97.66	97.37	97.37
5	96.49	94.74	97.66	97.37	97.66
Average (%)	96.37	94.85	97.54	97.6	**97.72**

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

Experiment #	Model		
	Neural network	RBFN with SCA	EQNFIS with quantum fuzzy entropy
1	9.5938	6.1094	6.0156
2	9.4531	6.0469	6.0781
3	9.5469	6.0625	5.8750
4	9.5156	6.0938	5.9688
5	9.5781	6.2344	5.9531
Average (second)	9.5375	6.1094	**5.9781**

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

Models	Average accuracy (%)
NNFS [17]	94.15
FEBFC [9]	95.14
NEFCLASS [13]	92.7
SANFIS [20]	96.3
MSC [12]	94.9
EQNFIS	**97.72**

98.54%. The means of re-substitution accuracy was 97.72%. The average classification accuracy of the EQNFIS model with quantum fuzzy entropy was better than that of other methods. Table 7 shows the CPU time of the cost of the EQNFIS model, the NN and RBFN. The average learning times of the EQNFIS, NN and RBFN were 5.9781, 6.1094 and 9.5375 s, respectively. We compared the testing accuracy of our model with that of other methods [9,12,13,17,20]. Table 8 shows the comparison between the learned EQNFIS models and other fuzzy logic system, neural network, and neuro-fuzzy classifiers. The average classification accuracy of the EQNFIS model was better than that of other methods.

5. Conclusion

In this paper, an entropy-based quantum neuro-fuzzy inference system (EQNFIS) was proposed for classification applications. The EQNFIS model is a five-layer structure, which combines the traditional Takagi–Sugeno–Kang (TSK). A self-constructing learning algorithm, which consists of the self-clustering algorithm (SCA), quantum fuzzy entropy, and the backpropagation algorithm, was also proposed. The advantages of the proposed EQNFIS model are summarized as follows: (1) it converges quickly; (2) it uses an online, fast, and one-pass self-constructing learning algorithm; (3) it has much lower rms error; and (4) it has a higher accuracy classification rate than other models. Finally, simulation results have shown that the average classification accuracy of the EQNFIS model is better than other methods. In addition to the simulations done in this paper, the proposed EQNFIS model has been used to solve face detection problem from color images in our laboratory.

Acknowledgement

This work was supported by National Science Council, R.O.C., under Grant no. NSC94-2218-E-324-004.

References

[1] P.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[2] L. Fei, Z. Shengmei, Z. Baoyu, Quantum neural network in speech recognition, Proc. IEEE Int. Conf. Signal Process. 6 (2000) 1267–1270.
[3] S. Halgamuge, M. Glesner, Neural networks in designing fuzzy systems for real world applications, Fuzzy Sets Syst. 65 (1994) 1–12.
[4] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man and Cybern. 23 (1993) 665–685.
[5] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Syst. 6 (1) (1998) 12–31.
[6] N. Kasabov, Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems, Fuzzy Sets Syst. 82 (1996) 135–149.
[7] R. Kretzschmar, R. Bueler, N.B. Karayiannis, F. Eggimann, Quantum neural networks versus conventional feedforward neural networks: an experimental study, Proc. IEEE Int. Conf. Signal Process. 1 (2000) 328–337.
[8] H.M. Lee, A neural network classifier with disjunctive fuzzy information, Neural Networks 11 (6) (1998) 1113–1125.
[9] H.M. Lee, C.M. Chen, J.M. Chen, Y.L. Jou, An efficient fuzzy classifier with feature selection based on fuzzy entropy, IEEE Trans. Syst., Man Cybern. B 31 (2001) 426–432.
[10] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks 6 (6) (1993) 861–867.
[11] C.J. Lin, C.T. Lin, An ART-based fuzzy adaptive learning control network, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 477–496.
[12] B.C. Lovel, A.P. Bradley, The multiscale classifier, IEEE Trans. Pattern Anal. Machine Intell. 18 (1996) 124–137.
[13] D. Nauck, R. Kruse, A neuro-fuzzy method to learn fuzzy classification rules from data, Fuzzy Sets Syst. 89 (1997) 277–288.
[14] S. Paul, S. Kumar, Subsethood-product fuzzy neural inference system (SuPFuNIS), IEEE Trans. Neural Networks 13 (3) (2002) 578–599.
[15] G. Purushothaman, N.B. Karayiannis, Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks, IEEE Trans. Neural Networks 8 (3) (1997) 679–693.
[16] M. Russo, FuGeNeSys—A fuzzy genetic neural system for fuzzy modeling, IEEE Trans. Fuzzy Syst. 6 (1998) 373–388.
[17] R. Setiono, H. Liu, Neural-network feature selector, IEEE Trans. Neural Network 8 (3) (1997) 654–662.
[18] P.K. Simpson, Fuzzy min-max neural networks-Part I: Classification, IEEE Trans. Neural Networks, 3 (1992) 776–786.
[19] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. SMC-15 (1985) 116–132.
[20] J.S. Wang, C.S. George Lee, Self-adaptive neuro-fuzzy inference systems for classification applications, IEEE Trans. Fuzzy Syst. 10 (6) (2002) 790–802.
[21] L.X. Wang, J.M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Sys. Man Cybern. 22 (6) (1992) 1414–1427.

[22] T.P. Wu, S.M. Chen, A new method for constructing membership functions and fuzzy rules from training examples, IEEE Trans. Syst. Man. Cybern. B 29 (1999) 25–40.

Cheng–Jian Lin received the B.S. degree in electrical engineering from Ta-Tung University, Taiwan, R.O.C., in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, R.O.C., in 1991 and 1996. From April 1996 to July 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan-Kai College, Nantou, Taiwan, R.O.C. Since August 1999, he has been with the Department of Computer Science and Information Engineering, Chaoyang University of Technology. Currently, he is a Professor of Computer Science and Information Engineering Department, Chaoyang University of Technology, Taichung, Taiwan, R.O.C. He served as the chairman of Computer Science and Information Engineering Department from 2001 to 2005. His current research interests are neural networks, fuzzy systems, pattern recognition, intelligence control, bioinformatics, and FPGA design. He has published more than 100 papers in the referred journals and conference proceedings. Dr. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the IEICE (The Institute of Electronics, Information and Communication Engineers), and the IEEE Computational Intelligence Society. He is an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI). Dr. Lin currently serves as the Associate Editor of International Journal of Applied Science and Engineering.

I–Fang Chung received the B. S. and M. S. degrees in control engineering from the National Chiao-Tung University (NCTU), Taiwan, in 1993 and 1995, respectively. He received the Ph.D. degree in Electrical and Control Engineering from NCTU in 2000. From 2000 to 2003, he was a Research Assistant Professor in Electrical and Control Engineering, NCTU. During 2003 to 2004, he worked as a postdoctoral fellow in the Institute of Medical Science, the laboratory of DNA information analysis of Human Genome Center of Tokyo University in Japan. Since 2004, he has served an assistant professor at the Institute of Bioinformatics, National Yang-Ming University, Taiwan. His current research interests are bioinformatics, machine learning, biomedical engineering, biomedical signal processing, and fuzzy neural networks.

Cheng–Hung Chen was born in Kaohsiung, Taiwan, R.O.C. in 1979. He received the B.S. and M.S. degree in computer science and information engineering from the Chaoyang University of Technology, Taiwan, R.O.C., in 2002 and 2004. He is currently pursuing the Ph.D. degree in electrical and control engineering from the National Chiao-Tung University, Taiwan, R.O.C. His current research interests are neural networks, fuzzy logic systems, intelligence control and pattern recognition.