# Quantum Neural Networks (QNN's): Inherently Fuzzy Feedforward Neural Networks

Gopathy Purushothaman and Nicolaos B. Karayiannis, *Member, IEEE*

***Abstract*— This paper introduces quantum neural networks (QNN's), a class of feedforward neural networks (FFNN's) inherently capable of estimating the structure of a feature space in the form of fuzzy sets. The hidden units of these networks develop quantized representations of the sample information provided by the training data set in various graded levels of certainty. Unlike other approaches attempting to merge fuzzy logic and neural networks, QNN's can be used in pattern classification problems without any restricting assumptions such as the availability of *a priori* knowledge or desired membership profile, convexity of classes, a limited number of classes, etc. Experimental results presented here show that QNN's are capable of recognizing structures in data, a property that conventional FFNN's with sigmoidal hidden units lack.**

***Index Terms*— Fuzzy classification, multilevel partitions, multilevel transfer functions, quantum neural networks, quantum neurons, uncertainty.**

## I. Introduction

FEEDFORWARD neural networks (FFNN's) have been a natural choice as trainable pattern classifiers and adaptive controllers because of their function approximation capability and generalization ability [4], [7], [9], [12], [18], [20], [26]. The function approximation capability allows them to form arbitrary nonlinear discriminant surfaces while the generalization ability allows them to respond consistently to data they were not trained with. These properties have resulted in FFNN classifiers and controllers being used in many applications [7], [15], [18].

In order to perform successfully in complex environments where many ill-defined and uncertain factors are encountered, human reasoning employs linguistic hedges, rules-of-thumb experience, intuition, and other heuristics [32]. Fuzzy inference systems such as fuzzy controllers and fuzzy pattern classifiers designed to incorporate these aspects of human reasoning in a qualitative manner have found many successful applications in pattern recognition [2], [14], [19], control [27], and other practical prediction and inferencing problems [29].

Many recent studies have focussed on developing systems capable of operating in complex uncertain environments by merging fuzzy and neural-network techniques [2], [10], [11], [13], [16]–[18], [20], [28]. Such hybrid systems can be classified into two classes: *Class 1*) Fuzzy inference systems that incorporate the learning and generalization abilities of FFNN's by using the FFNN paradigm within a general computational scheme [2], [8], [13] and *Class 2*) FFNN's that incorporate the imprecision and linguistic data handling abilities of fuzzy systems by using fuzzy set-theoretic learning rules [2], [10], [11], [16], [20], [28]. While the second class of hybrid systems possess the main advantages of the FFNN paradigm like function approximation ability, these properties have been proved only in some cases for Class 1 implementations. Even in these cases, some restrictions apply; for example, the *premise* may be required to be convex for the function approximation property to be true [13]. Thus the very flexible functionality of FFNN's is to some extent lost in these implementations. Nevertheless, Class 1 implementations are useful for identifying a complete fuzzy inference system because in Class 2 implementations the different modules of a fuzzy inference system (such as the premise, the rule-base, the defuzzification or output function, etc.) cannot be identified separately. This is due to the distributed nature of processing in FFNN's [7], [26].

Class 2 implementations could be useful in applications that do not require the identification of the subsystems of a fuzzy inference system but demand good generalization, such as pattern classification. This dichotomy between interpretability and generalization ability arises due to differences in the goals of modeling [3]. Moreover, a Class 2 implementation can also be used as part of a Class 1 implementation. As an example, an FFNN trained to generate the premise can be used as the first stage of any fuzzy inferencing system. In the rest of this paper, only Class 2 implementations are considered and, therefore, the discussion is restricted to pattern classifiers.

The first step in studying Class 2 hybrid systems is to analyze to what extent a typical FFNN trained in the conventional manner has the ability to function as a (Class 2) fuzzy system by itself. Suppose an FFNN is trained with example data that assign incremental regions of the feature space to one particular class only. In practical situations, such training sets may contain overlapping classes of data. The fundamental question is whether the trained FFNN is capable of identifying by itself the uncertainty present in the training set by forming gradual or "fuzzy" boundaries between the classes instead of forming sharp or "crisp" boundaries that divide the feature space into disjoint areas. In this context, several studies have investigated the ability of FFNN's trained with exemplars to generalize and divide the feature space with gradual or "fuzzy" boundaries instead of sharp or "crisp" boundaries [2], [10], [11], [16], [18], [20]–[23], [28]. These investigations have revealed the apparent limitations of FFNN's for detecting and

Manuscript received December 8, 1995; revised August 12, 1996 and December 22, 1996.

The authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4793 USA.

Publisher Item Identifier S 1045-9227(97)02765-3.

identifying the uncertainty present in a training set. In order to overcome these limitations, several approaches have been taken to design FFNN classifiers capable of partitioning a feature space consisting of overlapping classes of data with imprecise boundaries. Keller and Hunt [16] incorporated fuzzy logic principles into the development of learning algorithms for FFNN classifiers. The input to the network was formed by the feature vectors of the training data set. The "desired output" vector was computed from the membership functions chosen heuristically for each pattern class. These membership functions give a measure of the extent to which a certain training pattern belongs to each class. Pal and Mitra [20] trained multilayered FFNN's to function as fuzzy classifiers for data sets consisting of overlapping classes of data. The input vector to the network was formed by mapping the actual $n$-dimensional feature vector to a $3n$-dimensional space, with each component of the actual feature vector being mapped to three fuzzy sets representing *high*, *medium*, and *low* values in its domain. This allowed the network to handle uncertain information as well as to take linguistic inputs. The "desired output" for training the network was computed from membership functions chosen *a priori* for each pattern class. In both these training schemes, the membership functions have to be estimated or chosen *a priori*. Therefore in these schemes the trained network is capable of handling imprecise (and linguistic) inputs, but does not necessarily perform an estimation (generalization) task. However, model-free estimation of the input–output relationship is one of the main imports of the neural processing paradigm [2], [18]. Takagi and Hayashi [28] used FFNN's *per se* as estimators of fuzziness. In their approach, the training data were preclustered and labeled for supervised training. Therefore, this approach may not be valid when the classes of data are closely spaced or overlap in practical situations [6].

In summary, fuzzy FFNN classifiers are developed in the existing approaches either by explicitly training FFNN's to learn membership values estimated *a priori*, or by training FFNN's in the conventional manner and interpreting the response of the FFNN as being fuzzy in itself. The former approaches do not necessarily exploit the generalization ability of FFNN's, while the latter approaches assume that FFNN's are inherently fuzzy classifiers. In order to overcome these problems, Ishibuchi and Tanaka [11] have proposed an approximate pattern classification method for two-class discrimination problems based on their possibility–necessity analysis [10]. In this approach, each feature vector is classified as either *necessarily* or *possibly* belonging to a class. In other words, the fuzzy boundary generated by their approximate classifier encodes only two levels of uncertainty. Further, their approach uses two independent FFNN's and combines their responses to solve a two-class discrimination problem.

This paper introduces the *quantum neural network* (QNN), a computational tool for fuzzy classification that combines the advantages of neural modeling and fuzzy-theoretic principles. The salient features of the QNN are the following.

1) The ability to autonomously detect the presence of uncertainty in the sample data and adaptively learn to quantify the existing uncertainty. If the feature vector lies in the boundary between overlapping classes, the QNN will assign it partially to all classes whose overlapping boundaries include this feature vector. If there is no uncertainty regarding the classification of a certain feature vector, the QNN will assign it to the class indicated by the training set.
2) The ability to approximate any membership profile arbitrarily well from the sample data. The QNN does not depend on *a priori* knowledge or a desired membership profile that are required by most existing methods. The ability of feedforward neural models to approximate unknown functions is exploited in rendering this classifier capable of estimating by itself the uncertainty in the sample data in terms of membership values.
3) The ability to detect and quantify uncertainty in the sample data without any restricting assumptions about the number of classes, the number of uncertainty levels in the data, the convexity of the classes, etc.

Section II briefly reviews a recent theoretical investigation which showed the limitations of FFNN's for fuzzy classification [24]. Section III introduces the concept of multilevel partitioning of the feature space and proposes QNN's as an alternative neural architecture capable of generating multilevel partitions between any number of classes. Section IV presents learning algorithms for updating the internal parameters of QNN's. Section V presents an experimental evaluation of QNN's and compares their performance with that of conventional FFNN's. Section VI contains concluding remarks.

## II. Limitations of Conventional FFNN's for Fuzzy Classification

This section outlines the results of a previous rigorous theoretical study on the capacity of FFNN's for fuzzy classification [24]. Consider a network with $n_i$ inputs, $n_o$ output units, and one layer of $n_h$ hidden units. Let $\mathbf{v}_j^T = [v_{j1}, v_{j2}, \cdots, v_{jn_i}]$ be the weight vector connecting the $j$th hidden unit to the inputs and $\mathbf{w}_i^T = [w_{i1}, w_{i2}, \cdots, w_{in_h}]$ be the weight vector connecting the $i$th output unit to the hidden units. Let $\mathbf{V}$ be the matrix with the vectors $\mathbf{v}_j$ as its columns and $\mathbf{W}$ the matrix with the vectors $\mathbf{w}_i$ as its columns. Let the transfer function of the hidden units be the sigmoidal function $g : \mathcal{R} \rightarrow [0,1]$. A conventional FFNN may be defined as the function $\mathcal{N} : \mathcal{R}^{n_i} \rightarrow \mathcal{R}^{n_o}$, which maps $\mathbf{x} = [x_1, x_2, \cdots, x_{n_i}]^T$ to $\mathcal{N}(\mathbf{V}, g, \mathbf{W}; \mathbf{x})$, such that

$$\Pi_i \mathcal{N}(\mathbf{V}, g, \mathbf{W}; \mathbf{x}) = \sum_{j=1}^{n_h} w_{ij} g\left(\sum_{l=1}^{n_i} v_{jl} x_l + v_{j0}\right) + w_{i0} \quad \forall i$$

where $v_{j0}, w_{i0} \in \mathcal{R}, \forall j, i$, and $\Pi_i$ is the $i$th coordinate function $\Pi_i : \mathcal{R}^{n_o} \rightarrow \mathcal{R}$.

Consider a data set of $m$ feature vectors $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is a compact metric subspace of $\mathcal{R}^{n_i}$. Suppose $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{n_o} \subset \mathcal{X}$ are $n_o$ known classes of feature vectors in the feature space $\mathcal{X}$. Conventionally, FFNN's are trained using sample data to function as pattern classifiers by giving feature vectors belonging to the $i$th class as the input and adjusting the weights to obtain a response of "1" from the $i$th output unit and a response of "0" from the other output units. This procedure is

repeated for all the feature vectors belonging to all the classes in one *training epoch*.

The performance of the FFNN classifier trained in this manner is clearly influenced by the training algorithm and its architecture. Suppose the FFNN is trained to assign each feature vector to only one class. If the training algorithm is efficient and the architecture is optimal (i.e., it has a sufficient number of hidden units), then the FFNN will learn a close approximation to a discriminant function which minimizes the number of incorrect classifications [4], [9], [12], [25]. The discriminant function learned by the FFNN trained conventionally from sample data defines the boundaries between the classes of the feature space. In a recent study, the authors attempted to quantify how sharp or "crisp" these boundaries are [24]. In practice, an FFNN is not required to be trained till the output values saturate, i.e., till they become "1" or "0" [10], [16], [20], [28]. Therefore, an FFNN has learned to *consistently partition the feature space* of a given data set if for $i = 1, 2, \cdots, n_o$, $\Pi_i \mathcal{N}(\mathbf{x}) \in [0, 1]$ and $\Pi_i \mathcal{N}(\mathbf{x}) > \gamma$ if $\mathbf{x} \in \mathcal{C}_i$, $\Pi_i \mathcal{N}(\mathbf{x}) \leq \gamma$ otherwise, where $\gamma \in (0, 1)$. Usually, for symmetry and simplicity, the value of $\gamma$ may be taken to be 0.5. Suppose the FFNN is trained until it satisfies the condition

$$\mathcal{L}_1(\mathcal{N}, \mathcal{X}) = \frac{1}{mn_o} \sum_{i=1}^{n_o} \left[ \frac{1}{(1-\gamma)} \sum_{\mathbf{x}:\mathbf{x} \in \mathcal{C}_i} (\Pi_i \mathcal{N}(\mathbf{x}) - \gamma) + \frac{1}{\gamma} \sum_{\mathbf{x}:\mathbf{x} \notin \mathcal{C}_i} (\gamma - \Pi_i \mathcal{N}(\mathbf{x})) \right] \geq \eta \tag{1}$$

for some $\eta \in [0, 1]$. It is obvious that if the network has learned to consistently partition the feature space, then it satisfies the condition (1). This termination criterion is based on a global measure of how well the network has learned to function as a classifier. The quantity $\mathcal{L}_1(\mathcal{N}, \mathcal{X})$ measures the "goodness-of-fit" the network achieves for the function it is trained to learn. The parameter $\eta$ is the *amount* of *training* that the network is subjected to in order to achieve this goodness-of-fit. The parameter $\eta$ comprehensively models the training and learning aspects of the FFNN classifier in the following manner: Suppose that in the ideal situation where the optimal architecture size can be estimated and an efficient training algorithm can be used, the value of $\eta$ is set to one. In this case, the training process is guaranteed to terminate in finite time and the FFNN learns the discriminant function that minimizes the number of incorrect classifications [25]. In practice, $\eta$ takes a value less than one in order to accommodate for the suboptimality of the architecture and deficiencies of the learning algorithm. Finally, if the value of $\eta$ is closer to zero, then the network is only minimally trained or not trained at all. In other words, the parameter $\eta$ is a quantitative measure of all design factors affecting the classifier performance.

The main result of the study shows that the topology of the feature space influences the ability of FFNN's to function as fuzzy classifiers in the following manner: In a given data set, let the feature vector $\mathbf{p}$ be in the $i$th class and the vector $\mathbf{q}$ not be in the $i$th class. Let $G_i$ be the gradient of the response of the $i$th output unit evaluated at some point between $\mathbf{p}$ and $\mathbf{q}$. Then it can be shown that the $L_2$ norm of this gradient has the theoretical lower bound given by

$$\|G_i\| \geq c \frac{\eta}{\|\mathbf{p} - \mathbf{q}\|} - M \tag{2}$$

where $c$ and $M$ are positive constants. Suppose that the FFNN is trained with $\eta > 0$. As $\|\mathbf{p} - \mathbf{q}\| \to 0$, i.e., the classes are coming closer together or their overlapping increases, the norm of the gradient increases. In other words, the network forms an increasingly sharp or "crisp" boundary as $\|\mathbf{p} - \mathbf{q}\| \to 0$. This implies that such an FFNN loses its ability to function as a fuzzy classifier for data sets consisting of overlapping or closely spaced classes of feature vectors.

## III. The Quantum Neural Network (QNN)

The FFNN creates its internal representations from the sample information provided by the training data. In the remainder of this paper, training from sample data means that if a training vector belongs to the $i$th class, the $i$th output unit is required to respond with "1" while the response of all the other output units is required to be "0." In order to function as fuzzy classifier, the FFNN must use the sample information as a mere reference for creating the internal representations. Thus, it should not encode the sample information *accurately* into the internal representations. Such an exact or faithful encoding of the sample information results in the FFNN memorizing the "crispness" in the training data set. But an *inherently* fuzzy architecture should be capable of generalizing the sample information into various graded levels of certainty over the entire feature space. This may be possible if the architecture is capable of creating graded internal representations from the sample information. The QNN is proposed in this section as an architecture capable of allowing the sample information to be encoded into certain levels (grades) of certainty/uncertainty only.

### A. Training FFNN's for Fuzzy Classification

Consider the feature space $\mathcal{X}$ consisting of $n_o$ classes. For the $i$th class, consider $K(i)$ subsets $I_j^i \in \mathcal{X}$, $\forall j = 1, 2, \cdots, K(i)$, such that $\bigcup_{j=1}^{K(i)} I_j^i = \mathcal{X}$. Suppose that all the feature vectors in the subset $I_j^i$ can be assigned to the $i$th class with approximately the same membership value. Then $K(i)$ represents the number of levels of uncertainty in assigning feature vectors to the $i$th class. Let $\gamma_1^i > \gamma_2^i > \cdots > \gamma_{K(i)}^i > \gamma_{K(i)+1}^i$ be a set of nonnegative real numbers for each $i$. A *multilevel partition* on the feature space $\mathcal{X}$ is defined as the collection of the $n_o$ discrete fuzzy sets

$$\tilde{\mathcal{P}}_i = \sum_{\mathbf{x}:\mathbf{x} \in \mathcal{X}} \mu_i(\mathbf{x})/\mathbf{x}, \quad \forall i = 1, 2, \cdots, n_o$$

with

$$\mu_i(\mathbf{x}) = \sum_{j=1}^{K(i)} \frac{\gamma_j^i + \gamma_{j+1}^i}{2} \delta_j^i(\mathbf{x})$$

where $\delta_j^i(\mathbf{x}) = 1$ if $\mathbf{x} \in I_j^i$ and $\delta_j^i(\mathbf{x}) = 0$ otherwise. In the above, $\mu_i(\mathbf{x})$ is the membership value of the feature vector $\mathbf{x}$ to the $i$th class $\mathcal{C}_i$. The value of $\mu_i(\mathbf{x})$ for the feature

vector $\mathbf{x}$ depends on which subset $I_j^i$ of the feature space it belongs to. For example, if $\mathbf{x}$ belongs to a subset $I_k^i$ which is a region of the feature space that lies far away from the center of the $i$th class, then $\mu_i(\mathbf{x}) = (\gamma_k^i + \gamma_{k+1}^i)/2$ is very small. A multilevel partition is therefore a particular type of membership function, where the membership of the feature vectors is discrete valued over the feature space. It can be shown that the class of multilevel partitions can approximate arbitrary real-valued membership functions to any degree of accuracy [21]. Finally, multilevel partitions may be considered as a generalization of the possibility–necessity classification scheme proposed by Ishibuchi and Tanaka [10].

When a conventional FFNN is trained with a sample data set, some termination criteria are used to test if learning is satisfactory. The termination criterion (1) presented in the previous section can be used to check if the FFNN has learned to consistently partition the feature space. In a similar manner, a conventional FFNN can be defined to have learned to *consistently multilevel partition the feature space* of a given data set if for each $i = 1, 2, \cdots, n_o$, and for $j = 1, 2, \cdots, K(i)$, $\Pi_i \mathcal{N}(\mathbf{x}) \geq \gamma_{2j}^i$ if $\mathbf{x} \in I_{2j-1}^i$ and $\Pi_i \mathcal{N}(\mathbf{x}) \leq \gamma_{2j}^i$ if $\mathbf{x} \in I_{2j}^i$. Then the following termination criterion can be used to test if the FFNN has learned to consistently multilevel partition the feature space [21]:

$$\frac{2}{mn_o} \sum_{i=1}^{n_o} \sum_{j=1}^{\frac{K(i)}{2}} \left\{ \sum_{\mathbf{x}:\mathbf{x} \in I_{2j-1}^i} \left( \frac{\Pi_i \mathcal{N}(\mathbf{x}) - \gamma_{2j}^i}{\gamma_{2j-1}^i - \gamma_{2j}^i} \right) + \sum_{\mathbf{x}:\mathbf{x} \in I_{2j}^i} \left( \frac{\gamma_{2j}^i - \Pi_i \mathcal{N}(\mathbf{x})}{\gamma_{2j}^i - \gamma_{2j+1}^i} \right) \right\} \geq \eta > 0. \quad (3)$$

Suppose a conventional FFNN learns to consistently partition the feature space of the data set $\mathcal{X}$ by satisfying the above termination criterion with the intervals $\gamma_j^i - \gamma_{j+1}^i = \theta$, $\forall i$ and $\forall j$. Let $(\mathbf{p}_{(j)}^{(i)}, \mathbf{q}_{(j)}^{(i)})$ be an ordered pair of feature vectors such that $\mathbf{p}_{(j)}^{(i)} \in I_j^i$ and $\mathbf{q}_{(j)}^{(i)} \notin I_j^i$. Then, from the definition of a multilevel partition above, $\mathbf{p}_{(j)}^{(i)}$ belongs to the class $\mathcal{C}_i$ with membership value $(\gamma_j^i + \gamma_{j+1}^i)/2$ and $\mathbf{q}_{(j)}^{(i)}$ belongs to $\mathcal{C}_i$ with a different membership value. Let the $\mathrm{L}_2$ norm $||\mathbf{p}_{(j)}^{(i)} - \mathbf{q}_{(j)}^{(i)}||$ be the distance of separation between the two feature vectors. Consider all pairs of feature vectors such that one feature vector belongs to the class $\mathcal{C}_i$ with membership value $(\gamma_j^i + \gamma_{j+1}^i)/2$, and the other belongs to $\mathcal{C}_i$ with a different membership value. Let $D_i = \max_j ||\mathbf{p}_{(j)}^{(i)} - \mathbf{q}_{(j)}^{(i)}||$ be the maximum distance of separation between all such pairs of feature vectors in the $i$th class. Then $D = \max_i D_i$ is the maximum distance of separation of all such pairs of vectors lying across the boundaries of all the classes in the given data set. Let $||G||$ be the $\mathrm{L}_2$ norm of the gradient of the network response evaluated at a point between $\mathbf{p}_{(j)}^{(i)}$ and $\mathbf{q}_{(j)}^{(i)}$ and averaged over all $i$ and $j$. Then it can be shown that $||G||$ has the theoretical lower bound given by

$$||G|| \geq \left( c_1 \frac{\eta - c_2}{D} \right) \theta - M \quad (4)$$

where $c_1$, $c_2$ and $M$ are positive constants [21], [24].

The parameter $\theta$ is independent of the amount of training $\eta$ and is dependent only on the fine structure of the "multilevel partitions" defined by the subsets $I_j^i$ and membership values $\gamma_j^i$. Thus inequality (4) shows that by independently varying $\theta$, the gradient of the network response can be kept as small as required. Therefore, it follows from (2) and (4) that fuzzy partitioning of the feature space can be achieved if the FFNN is trained to multilevel partition the feature space by satisfying the termination criterion (3) instead of being trained in the conventional manner, that is, by satisfying the termination criterion (1).

In practice, there is no guarantee that, in practice, an FFNN trained with sample data will satisfy the stringent termination criterion (3). This suggests the need for an alternative feedforward neural model that is capable of generating by itself a consistent multilevel partition of the feature space. This model is described in detail in the following section.

### B. Quantum Neuron

The aforementioned objectives can be accomplished by constructing a multilayered neural network so that whenever the network forms a partition between closely spaced classes, the partition truly represents the imprecision at the boundary between the classes. The previous section shows that the partitions between closely spaced classes must be at least multilevel, with the fine structure of the levels representative of the uncertainty present in the data lying at the boundary between these classes. One simple way of incorporating the ability to form consistent multilevel partitions in the hidden layer is to create hidden unit partitions with the property of "spreading-out" over regions of uncertainty in the feature space and "collapsing-in" over regions of certainty. If all the hidden unit transfer functions have the ability to form "graded" partitions instead of the crisp linear partitions, then these partitions can be "collapsed-in" or "spread-out" as required, using a suitable algorithm. Such an algorithm will not require that the fuzzy measures on the feature space be known, but will be a general procedure for learning the imprecision and uncertainty in the data set. This motivates the study of hidden units with *multilevel* transfer functions.

Suppose the multilevel hidden unit has $n_s$ discrete *states* or *levels*. Then its transfer function can be written as a superposition of $n_s$ sigmoidal functions, each shifted by $\theta^r$. The output of this multilevel unit can be written as $(1/n_s) \sum_{r=1}^{n_s} \mathrm{sgm}(\mathbf{v}^T\mathbf{x} - \theta^r)$, where $\mathrm{sgm}(\cdot)$ is a sigmoid function. The step widths of the multilevel transfer function, which may be called the *quantum intervals*, will be representative of discrete localized cells in the feature space consisting of feature vectors with approximately the same level of uncertainty as to their membership to the classes in the data set. These quantum intervals are determined by the "jump-positions" $\theta^r$. Unlike the step widths, the step heights need not be learned through independent parameters. This is because several sigmoids can be shifted to the same location and added together to give steps of desired heights, to an approximation. This approximation reduces the total number of parameters to be learned by almost one-third. Finally, it will

be clear from Section V that in practice this approximation works well.

The QNN consists of $n_i$ inputs, one layer of $n_h$ multilevel hidden units, and $n_o$ output units. The output units can be linear or sigmoidal. Let the synaptic weight connecting the $i$th output unit to the $j$th hidden unit be $w_{ij}$. Let the synaptic weight connecting the $j$th hidden unit to the $l$th input be $v_{jl}$. Let $\mathbf{x}_k = [x_{1,k}, x_{2,k}, \cdots, x_{n_i,k}]^T,\ \forall k = 1, \cdots, m$, be the $m$ feature vectors of the data set $\mathcal{X}$. Then the input to the $j$th hidden unit from the $k$th feature vector is $\overline{h}_{j,k} = \sum_{l=0}^{n_i} v_{jl} x_{l,k}$ with $x_{0,k} = 1,\ \forall k$. Therefore, the response of the $j$th hidden unit for the $k$th feature vector can be written as

$$\tilde{h}_{j,k} = \frac{1}{n_s}\sum_{r=1}^{n_s} h_{j,k}^r = \frac{1}{n_s}\sum_{r=1}^{n_s} \mathrm{sgm}\left(\beta_h\left(\overline{h}_{j,k} - \theta_j^r\right)\right) \quad (5)$$

where $\mathrm{sgm}(\tau) = 1/(1+\exp(-\tau))$ is a sigmoid function, $\beta_h$ is a slope factor, $\theta_j^r$'s define the jump positions in the transfer function, and $n_s$ is the number of levels or sigmoids in the hidden unit. Fig. 1(a) plots the response $\tilde{h}_{j,k}$ of a four-level quantum neuron as a function of its input $\overline{h}_{j,k}$. Fig. 1(b) demonstrates the generation of unequal step heights through simple shifting. Each $\tilde{h}_{j,k}$ is one elemental fuzzy partition. Similarly, the input to the $i$th output unit from the $k$th feature vector is $\overline{y}_{i,k} = \sum_{j=0}^{n_h} w_{ij}\tilde{h}_{j,k}$ with $\tilde{h}_{0,k} = 1,\ \forall k$. Therefore, the response of the $i$th output unit for the $k$th feature vector can be written as

$$\hat{y}_{i,k} = \begin{cases} \mathrm{sgm}(\beta_o(\overline{y}_{i,k})) & \text{if the } i\text{th output unit is sigmoidal} \\ \overline{y}_{i,k} & \text{if the } i\text{th output unit is linear.} \end{cases} \quad (6)$$

The essential difference between adding more sigmoidal hidden units and having multilevel hidden units is clear from (5). The linear partition generated by an additional hidden unit has all the degree of freedom to align itself along any direction on the feature space. On the other hand, the sigmoids within multilevel hidden unit transfer function can only "spread-out" or "collapse-in" parallel to each other. This difference is further explained in Section V.

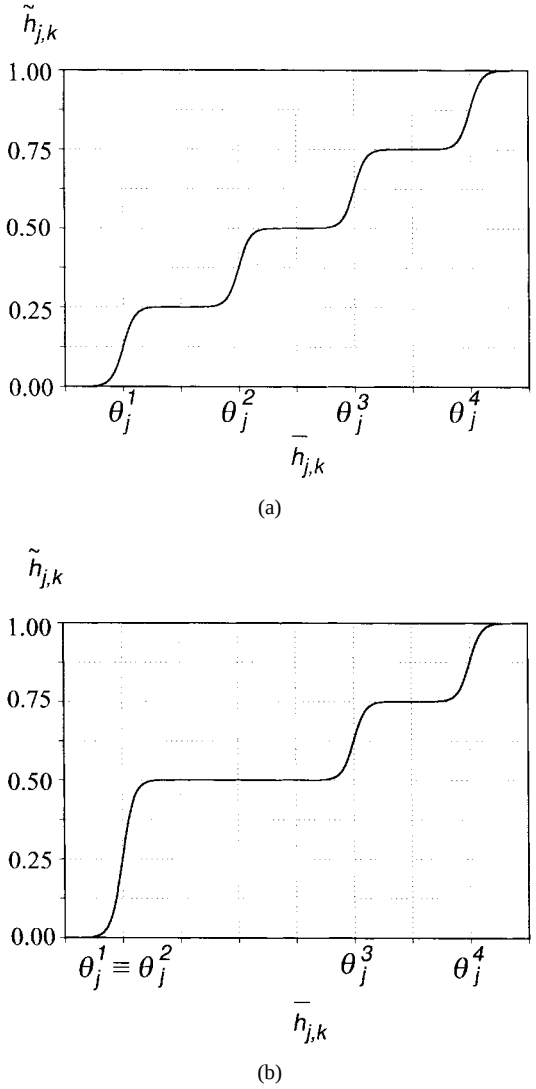



Fig. 1. A multilevel transfer function with (a) equal step heights (b) unequal step heights.

## IV. A Gradient-Descent-Based Learning Algorithm for the QNN

The learning of the QNN parameters is considered in two steps. The synaptic weights need to be updated first in order to train the QNN to consistently partition the feature space of the given data set. Simultaneously, the uncertainty present in the feature space must be learned through the adaptation of the parameters $\theta_j^r$.

### A. Updating the Weights in QNN

Let $\mathbf{y}_k = [y_{1,k}\ y_{2,k} \cdots\ y_{n_o,k}]^T$ be the desired output vector for the $k$th input feature vector $\mathbf{x}_k$, where $y_{i,k} = 1$ if $\mathbf{x}_k \in \mathcal{C}_i$ and $y_{i,k} = 0$ if $\mathbf{x}_k \notin \mathcal{C}_i$. Let $\hat{\mathbf{y}}_k = [\hat{y}_{1,k}\ \hat{y}_{2,k} \cdots\ \hat{y}_{n_o,k}]^T$ be the actual output vector. A gradient-descent-based algorithm for learning the synaptic weights of the QNN can be derived by minimizing the quadratic error function

$$E_k = \frac{1}{2}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2 \quad (7)$$

sequentially for each $k$ [15]. Each feature vector $\mathbf{x}_k$ is given as the input to the QNN and the synaptic weights are adjusted so that $E_k$ is minimized. This can be achieved by *adapting* or changing each synaptic weight by an amount proportional to the gradient of $E_k$ with respect to that particular synaptic weight [15]. The update equation for the synaptic weight $w_{pj}$ connecting the $p$th output unit to the $j$th hidden unit is derived in Appendix A as

$$w_{pj,k} = w_{pj,k-1} + \alpha\epsilon_{p,k}^o\tilde{h}_{j,k} \quad (8)$$

where $w_{pj,k-1}$ and $w_{pj,k}$ are the values of $w_{pj}$ before and after the adaptation for the $k$th input, $\alpha$ is the learning rate and

$$\epsilon_{p,k}^o = \begin{cases} \beta_o\hat{y}_{p,k}(1-\hat{y}_{p,k})(y_{p,k}-\hat{y}_{p,k}) & \text{if } \hat{y}_{p,k} = \mathrm{sgm}(\beta_o(\overline{y}_{p,k})) \\ (y_{p,k}-\hat{y}_{p,k}) & \text{if } \hat{y}_{p,k} = \overline{y}_{p,k}. \end{cases} \quad (9)$$

The update equation for the synaptic weight $v_{sq}$ connecting the $s$th hidden unit to the $q$th input is derived in Appendix A as

$$v_{sq,k} = v_{sq,k-1} + \alpha\beta_h\epsilon_{s,k}^h x_{q,k} \quad (10)$$

where $v_{sq,k-1}$ and $v_{sq,k}$ are the values of $v_{sq}$ before and after the adaptation, $\alpha$ is the learning rate, and

$$\epsilon^h_{s,k} = \left(\frac{1}{n_s}\sum_{r=1}^{n_s} h^r_{s,k}\left(1 - h^r_{s,k}\right)\right)\sum_{i=1}^{n_o}\epsilon^o_{i,k} w_{is}. \qquad (11)$$

### B. Updating the Quantum Intervals

The QNN must be first trained to recognize the occurrence of transitions between classes. The synaptic weights of the QNN must be updated to enable the network to learn the class boundaries on the feature space. At this point, the hidden unit partitions may not have a one-to-one correspondence with the decision space partitions. Some hidden unit partitions may even pass right through the "central tendencies" of classes. However, each decision space boundary is a linear superposition of many hidden unit partitions, partially or completely. The objectives of this algorithm are 1) to selectively "collapse-in" the multilevel transfer functions of the hidden units whose partitions do not form class boundaries or pass through the central tendencies of classes and 2) to selectively "spread-out" the transfer functions of the hidden units whose partitions lie in the neighborhood of a class boundary [24].

It is proposed in this paper that the quantum intervals can be learned by minimizing the *class-conditional variances* [6] at the outputs of the hidden units. It is expected that such updating of $\theta^s_q$ will help create steps of different heights according to the concentration of feature vectors belonging to different classes lying at the class boundaries.

The variance of the output of the $p$th hidden unit for the $m$th class $\mathcal{C}_m$ is given by

$$\sigma^2_{p,m} = \sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\left(\langle\tilde{h}_{p,\mathcal{C}_m}\rangle - \tilde{h}_{p,k}\right)^2$$

where

$$\langle\tilde{h}_{p,\mathcal{C}_m}\rangle = \frac{1}{|\mathcal{C}_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\tilde{h}_{p,k} \qquad (12)$$

and $|\mathcal{C}_m|$ denotes the cardinality of $\mathcal{C}_m$. The adaptation of the parameters $\theta^s_q$ is based on the minimization of the objective function formed by summing $\sigma^2_{p,m}$ over all the classes and all the hidden units, i.e.,

$$G = \frac{1}{2}\sum_{p=1}^{n_h}\sum_{m=1}^{n_o}\sigma^2_{p,m} = \frac{1}{2}\sum_{p=1}^{n_h}\sum_{m=1}^{n_o}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\left(\langle\tilde{h}_{p,\mathcal{C}_m}\rangle - \tilde{h}_{p,k}\right)^2. \qquad (13)$$

The update equation for $\theta^s_q$ is derived in Appendix B as

$$\Delta\theta^s_q = \alpha_\theta\frac{\beta_h}{n_s}\sum_{m=1}^{n_o}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\left(\langle\tilde{h}_{q,\mathcal{C}_m}\rangle - \tilde{h}_{q,k}\right) \times\left(\langle\nu^s_{q,\mathcal{C}_m}\rangle - \nu^s_{q,k}\right) \qquad (14)$$

where $\alpha_\theta$ is the learning rate, $\langle\nu^s_{q,\mathcal{C}_m}\rangle = \frac{1}{|\mathcal{C}_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\nu^s_{q,k}$, and $\nu^s_{q,k} = h^s_{q,k}(1 - h^s_{q,k})$.

TABLE I
ALGORITHM FOR TRAINING THE QNN

Select $\alpha$, $\alpha_\theta$, $\beta_h$, $\beta_o$.
Randomly initialize the weights and $\theta^r_j$ for $j = 1,2,\ldots,n_h$ and $r = 1,2,\ldots,n_s$.
Update the synaptic weights:
  For $k = 1,2,\ldots,m$ :
    For $j = 1,2,\ldots,n_h$ :
      $h_{j,k} \leftarrow \sum_{i=0}^{n_i} v_{ji}\, x_{i,k}$
      $h^r_{j,k} \leftarrow \sum_{r=1}^{n_s}\mathrm{sgm}(\beta_h(\bar{h}_{j,k} - \theta^r_j))$
      $\tilde{h}_{j,k} \leftarrow \frac{1}{n_s}\sum_{r=1}^{n_s} h^r_{j,k}$
    For $i = 1,2,\ldots,n_o$ :
      $y_{i,k} \leftarrow \sum_{j=0}^{n_h} w_{ij}\,\tilde{h}_{j,k}$
      $\hat{y}_{i,k} \leftarrow \begin{cases}\mathrm{sgm}(\beta_o(\bar{y}_{i,k})) & \text{if the } i\text{th output unit is sigmoidal}\\ \bar{y}_{i,k} & \text{if the } i\text{th output unit is linear}\end{cases}$
    For $i = 1,2,\ldots,n_o$ :
      $\epsilon^o_{i,k} \leftarrow \begin{cases}(y_{i,k} - \hat{y}_{i,k})\hat{y}_{i,k}(1 - \hat{y}_{i,k}) & \text{if the } i\text{th output unit is sigmoidal}\\ (y_{i,k} - \hat{y}_{i,k}) & \text{if the } i\text{th output unit is linear}\end{cases}$
    For $j = 1,2,\ldots,n_h$ and $i = 1,2,\ldots,n_o$ :
      $w_{ij} \leftarrow w_{ij} + \alpha\,\epsilon^o_{i,k}\,\tilde{h}_{j,k}$
    For $j = 1,2,\ldots,n_o$ :
      $\epsilon^h_{j,k} \leftarrow (\frac{1}{n_s}\sum_{r=1}^{n_s} h^r_{j,k}(1 - h^r_{j,k}))\sum_{i=1}^{n_o}\epsilon^o_{i,k} w_{ij}$
    For $j = 1,2,\ldots,n_h$ and $l = 1,2,\ldots,n_i$ :
      $v_{jl} \leftarrow v_{jl} + \alpha\,\beta_h\,\epsilon^h_{j,k}\,x_{l,k}$
Update the quantum intervals:
  For $k = 1,2,\ldots,m$ :
  For $j = 1,2,\ldots,n_h$ :
    $\bar{h}_{j,k} \leftarrow \sum_{l=0}^{n_i} v_{jl}\,x_{l,k}$
    $h^r_{j,k} \leftarrow \sum_{r=1}^{n_s}\mathrm{sgm}(\beta_h(\bar{h}_{j,k} - \theta^r_j))$
    $\tilde{h}_{j,k} \leftarrow \frac{1}{n_s}\sum_{r=1}^{n_s} h^r_{j,k}$
    $\nu^r_{j,k} \leftarrow h^r_{j,k}(1 - h^r_{j,k})$
  For $j = 1,2,\ldots,n_h$ :
    $\langle\tilde{h}_{j,\mathcal{C}_m}\rangle \leftarrow \frac{1}{|\mathcal{C}_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\tilde{h}_{j,k}$
    $\langle\nu^r_{j,\mathcal{C}_m}\rangle \leftarrow \frac{1}{|\mathcal{C}_m|}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}\nu^r_{j,k}$
  For $k = 1,2,\ldots,m$ :
  For $q = 1,2,\ldots,n_h$ and $r = 1,2,\ldots,n_s$ :
    $\theta^r_q \leftarrow \theta^r_q + \alpha_\theta\,\frac{\beta_h}{n_s}\sum_{m=1}^{n_o}\sum_{\mathbf{x}_k:\mathbf{x}_k\in\mathcal{C}_m}(\langle\tilde{h}_{q,\mathcal{C}_m}\rangle - \tilde{h}_{q,k})(\langle\nu^r_{q,\mathcal{C}_m}\rangle - \nu^r_{j,k})$

The network is trained in a sequence of *adaptation cycles*. Each adaptation cycle involves the adaptation of all the internal parameters of the network, that is, the synaptic weights and the locations $\theta^s_q$ of the shifted and superimposed sigmoid functions of the hidden units. Since the criterion employed for updating the parameters $\theta^s_q$ is based on all the input vectors from the training set, $\theta^s_q$ are updated after the presentation of all the inputs to the network and the corresponding adaptation of the synaptic weights. The algorithm is summarized in Table I.

The algorithms proposed in this paper are based on gradient descent. Nevertheless, any other algorithm that facilitates fast learning or minimizes the occurrence of local minima can also be used for updating the synaptic weights and the quantum intervals, according to necessity and the nature of the data sets [15]. Since the main aim of this paper is to demonstrate the capacity of the architecture, all the experiments in the following section were performed using the simple gradient-descent-based algorithms derived in this section. Therefore, the learning time or convergence rate in all the following experiments are mentioned as the number of training epochs required for the sum-squared error to reach a certain minimum value. These convergence rates are comparable to those of any other gradient-descent-based algorithm applied to similar problems [15].

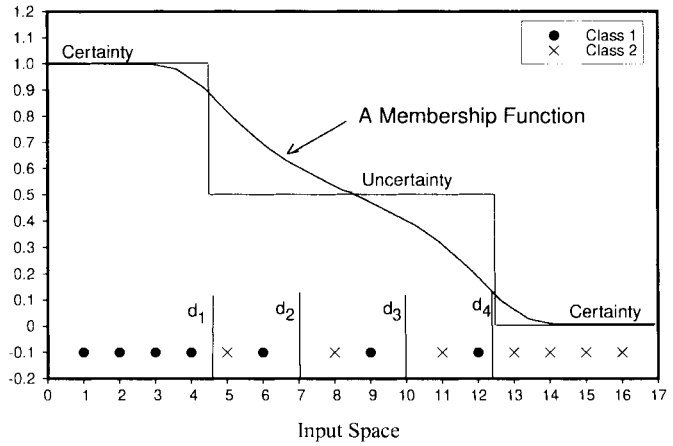



Fig. 2. A simple two-class data set.

## V. Experimental Results

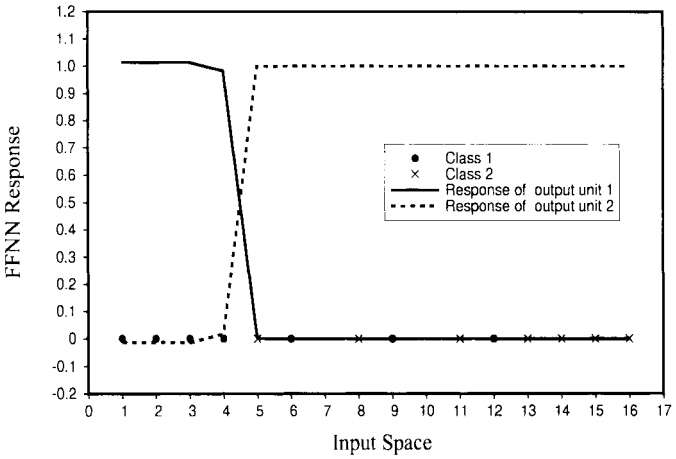
This section presents experiments designed to demonstrate that 1) the partitions of the feature space generated by conventional FFNN's are crisp and 2) the partitions of the feature space generated by QNN's are fuzzy and intuitive. This is achieved by explicitly viewing the decision spaces of the FFNN and the QNN. These experiments are performed with several data sets specifically chosen to clearly demonstrate these properties without compromising the complexity present in actual data sets.

### *A. A Simple Two-Class Data Set*

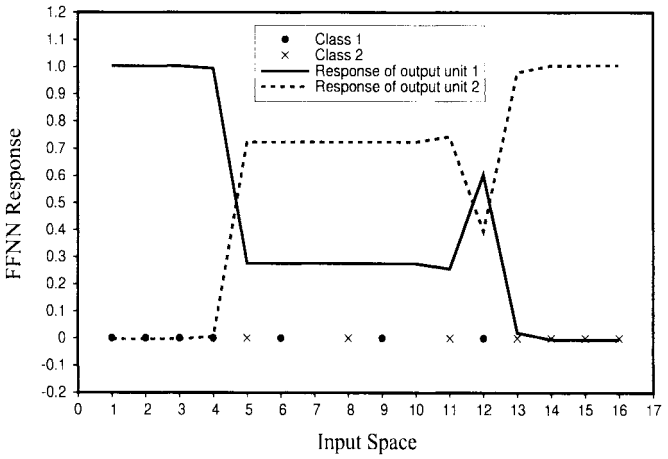
Consider the simple two-class data set shown in Fig. 2. This figure shows two classes of data sets represented as the "circles" and the "crosses." An intuitive interpretation of a normal physical object space whose features are represented in this data set will be the following: at the extreme left, maximum certainty that the data points belong to class 1; at the extreme right, maximum certainty that the data points are in class 2; in the middle, maximum uncertainty as to which class the data points belong to. A more refined description of the structure in this feature space is encoded into the membership function shown in Fig. 2.

In the first experiment, a Bayesian classifier was designed to minimize the probability of incorrect classification. This resulted in one of the four linear discriminant functions $d_i$, $i = 1, 2, 3, 4$ shown in Fig. 2, depending on the parameters of the distributions. For each of the resulting classifiers, the number of classification errors on this data set was the minimum possible value of three.

In the second experiment, a conventional FFNN was trained to function as the classifier for this data set. This FFNN was designed with one input, one hidden unit, and two output units. Though this design is highly redundant, it serves to explain the function of the network components with ease. The network was trained using the error backpropagation (EBP) algorithm [26], with a learning rate of $7\times10^{-2}$ and no momentum. The training was terminated when the sum-squared error reduced to $1\times10^{-2}$ or at the 100 000th cycle, whichever was earlier. The responses of both output units are shown in Fig. 3(a).




(a)



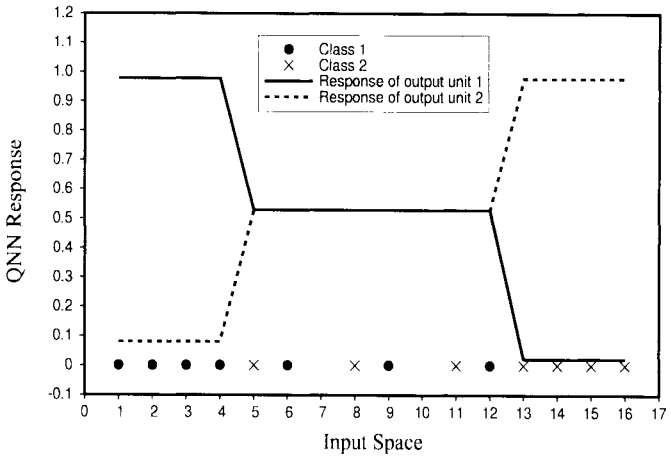

(b)

Fig. 3. Response of the FFNN classifier for the two-class data with (a) a single hidden unit and (b) three hidden units.

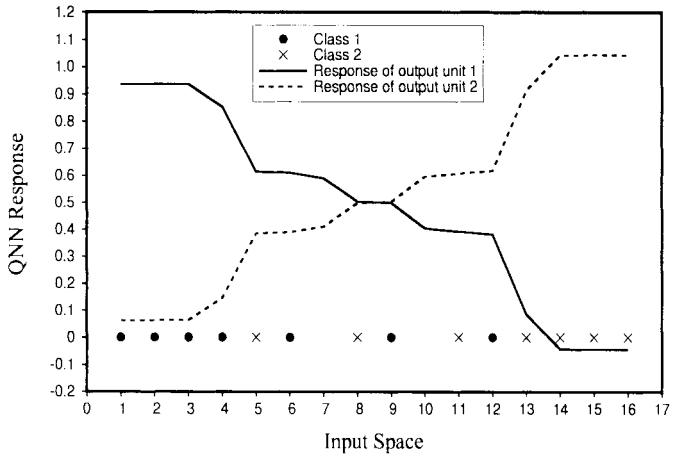
The following inferences can be made: The transition of the response curves from one state to the next occurs at the location of $d_1$ shown in Fig. 2. This is consistent with the well-known result that the conventional FFNN classifier is a minimum probability of error classifier [25]. Since *at* the transition point the feature vectors belonging to the two classes are close to each other, the transition is rather abrupt.

A conventional FFNN with one input, three hidden units, and two output units was also trained on this data set. The response of this FFNN is shown in Fig. 3(b). The response of the first output unit for the four "circles" in the extreme left is greater than 0.5. The response of the first output unit for the "circle" in the extreme right is also greater than 0.5. Therefore, the response of the first output unit is greater than 0.5 for these five circles. This FFNN misclassifies only the two circles located on the $X$-axis at 6 and 9. This experiment demonstrates that varying the number of hidden units only results in the overfitting of the data and does not have the desired effect of producing graded responses.

A QNN with one input, one multilevel hidden unit, and two output units was trained on the data set of Fig. 2. In the first experiment, a two-level hidden unit was used. The synaptic




(a)




(b)

Fig. 4. Response of the QNN classifier for the two-class data with (a) a two-level hidden unit and (b) a nine-level hidden unit.

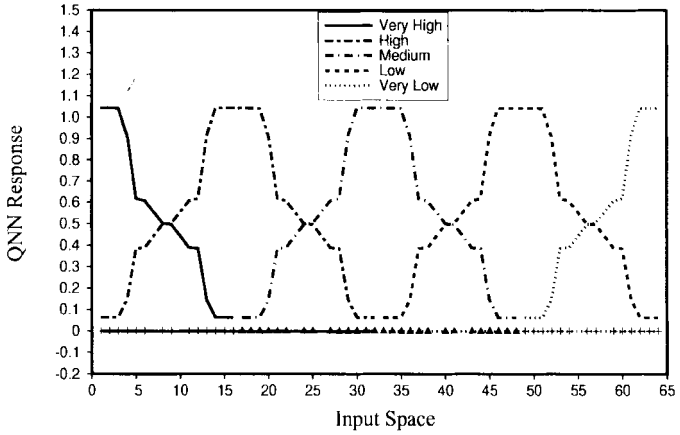



Fig. 5. Fuzzy sets of the feature space estimated by the QNN.

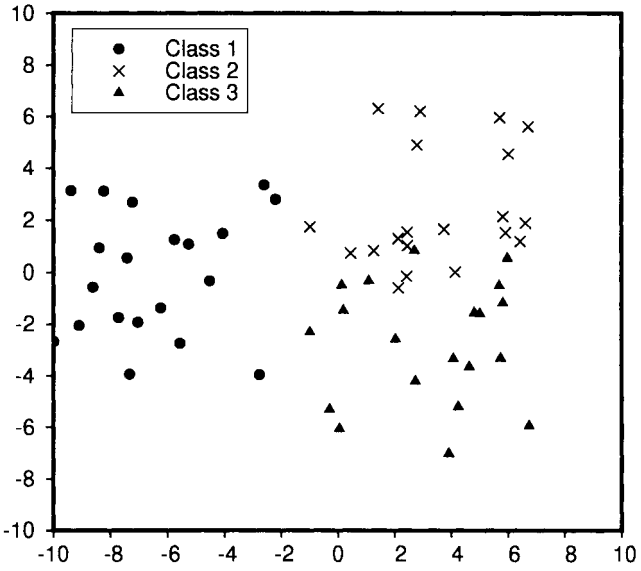



Fig. 6. A three-class hybrid data set.

weights in the network were updated using the algorithm derived in Section IV, with a learning rate of $7\times10^{-2}$. The hidden unit was tuned to learn the quantum intervals with a learning rate of $1\times10^{-2}$. In either case no momentum was provided. Fig. 4(a) shows the response of this QNN. These response curves are to be compared with the intuitive responses sketched in Fig. 2. Since only two levels were used in the hidden unit of this QNN, this response encodes only three degrees of certainty. In order to realize a finer representation of the imprecision in this data set, the next experiment was performed on a QNN with nine levels in the hidden unit. The response of this QNN is shown in Fig. 4(b). It is clear that if the QNN is expected to respond with many degrees of certainty, then the two extremes have to be assigned the two "very certain" categories. In the middle, corresponding to a maximum uncertainty, the network response has to be approximately 0.5. Fig. 4(b) clearly shows all these characteristics.

In Fig. 4(b), the QNN has divided the feature space into two fuzzy sets and has estimated the membership functions for these two sets. Often membership functions on the feature space of a data set are defined by dividing each component axis of the feature space into three fuzzy sets representing *high*, *medium*, and *low* values, or into a finer partition of five fuzzy sets representing *very high*, *high*, *medium*, *low*, and *very low* values [18], [19]. Consider a one-dimensional feature space as an extrapolation of the data set shown in Fig. 2, i.e., a feature space generated by periodically repeating the data set in Fig. 2. The response of the QNN with nine levels in the hidden unit is shown in Fig. 5. This figure shows the feature space divided into five fuzzy sets representing *very high*, *high*, *medium*, *low*, and *very low* values.

### *B. A Three-Class Hybrid Data Set*

Consider a three-class data set on a two-dimensional feature space shown in Fig. 6. This data set is "hybrid" because it consists of both overlapping and nonoverlapping classes of data. This means that the network receives information that is mixed, i.e., on one part of the feature space there is certainty as to the state of the nature while on another part there is uncertainty. The "circles" form the first class, the "crosses" form the second class and the "triangles" form the third class.

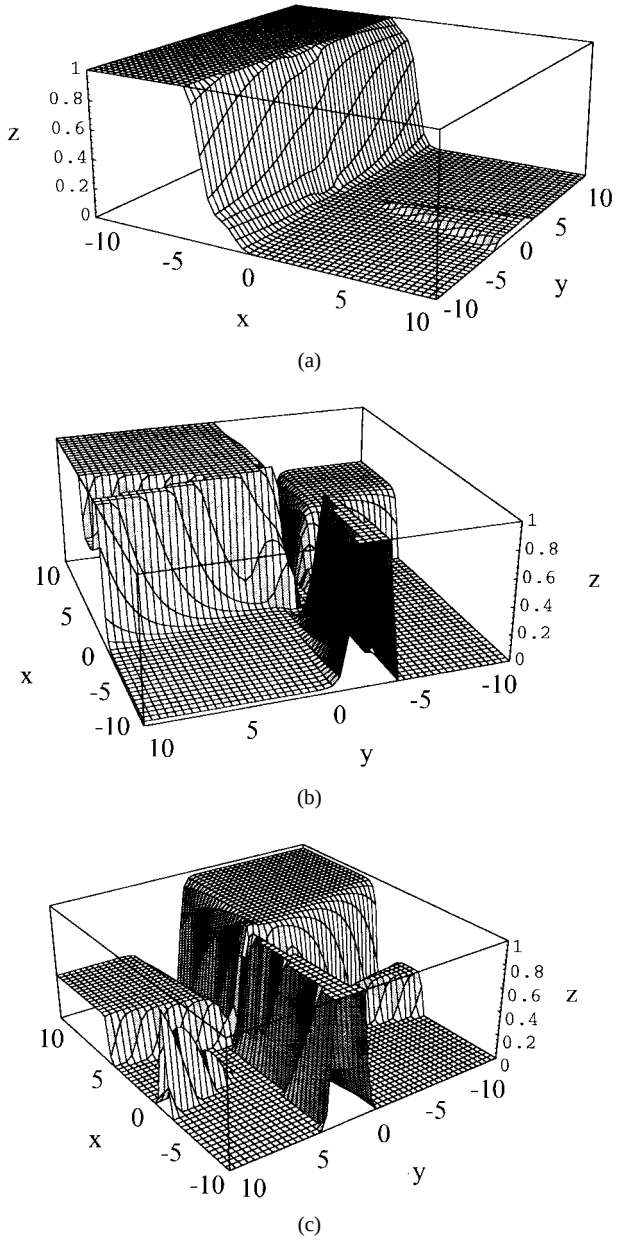



Fig. 7. Response of the FFNN for (a) the circles, (b) the crosses, and (c) the triangles.

Each class consists of 20 data points. The circles are well separated from the crosses and triangles. The crosses and the triangles have considerable overlapping among themselves. Again, an intuitive interpretation of this data set will be the following: The part of the feature space to the left of the plane $x = -2.0$ is predominantly "circles" and the part of the feature space to the right of this plane is clearly not "circles." The extreme top portion of the feature space to the right of $x = -2.0$ is certainly "crosses," and the extreme bottom region of this part of the feature space is certainly "triangles." However, toward the middle of the feature space in the plane $x > -2.0$, there is uncertainty as to whether this region represents the "crosses" or the "triangles." But, there is no uncertainty as to the fact that this region is not representative of "circles."

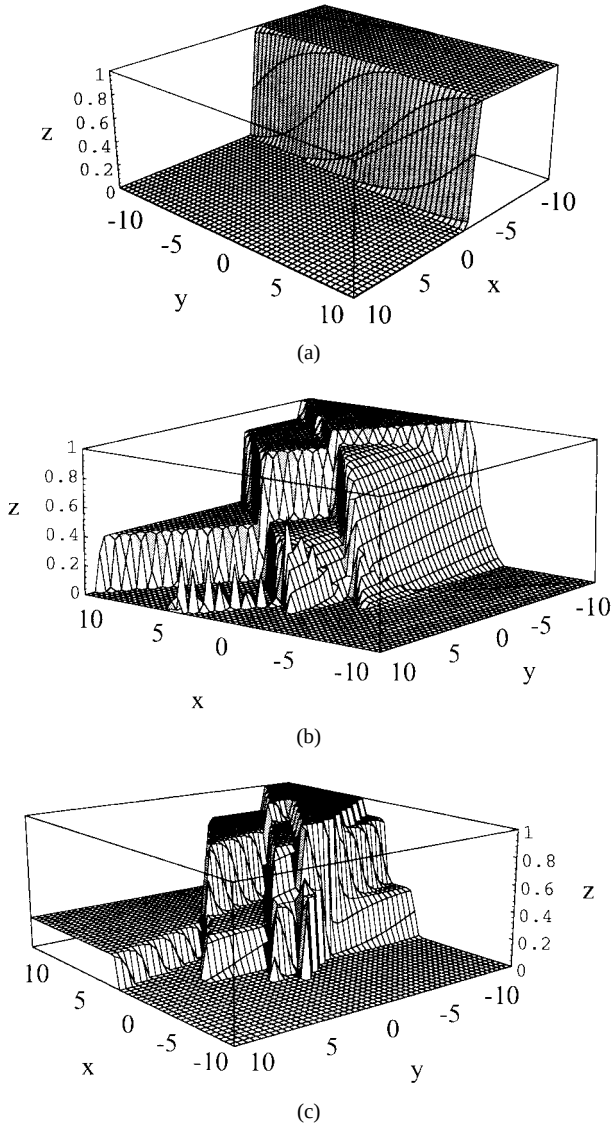



Fig. 8. Response of the QNN (a) with two-level hidden units for the circles, (b) with two-level hidden units for the triangles, and (c) with three-level hidden units for the traingles.

A conventional FFNN with two inputs and three output units was trained to consistently partition the feature space of this data set using the EBP with a learning rate of $1\times10^{-2}$. The number of hidden units were varied from three to eight. The FFNN with five hidden units was found to be satisfactory in terms of the minimum error achievable for a given number of adaptation cycles, and the overall time required for training the network. The decision spaces of this FFNN were plotted in three dimensions (3-D), by taking the two features as the $x$ and the $y$ axes and the output of the network as the $z$ axis. The decision space formed by the first output unit of the FFNN is shown in Fig. 7(a). The graduated $z$ axis in Fig. 7(a) is located at $(-10,-10)$ on the feature space. It is clear that the response of the FFNN is consistent with the distribution of the data on the feature space. The plane $x > -2.0$ does not contain any circles so the response of the first unit over this plane is almost zero everywhere. Over the plane $x \leq -2.0$, the response of the first output unit is almost 1.0 everywhere. The transition from

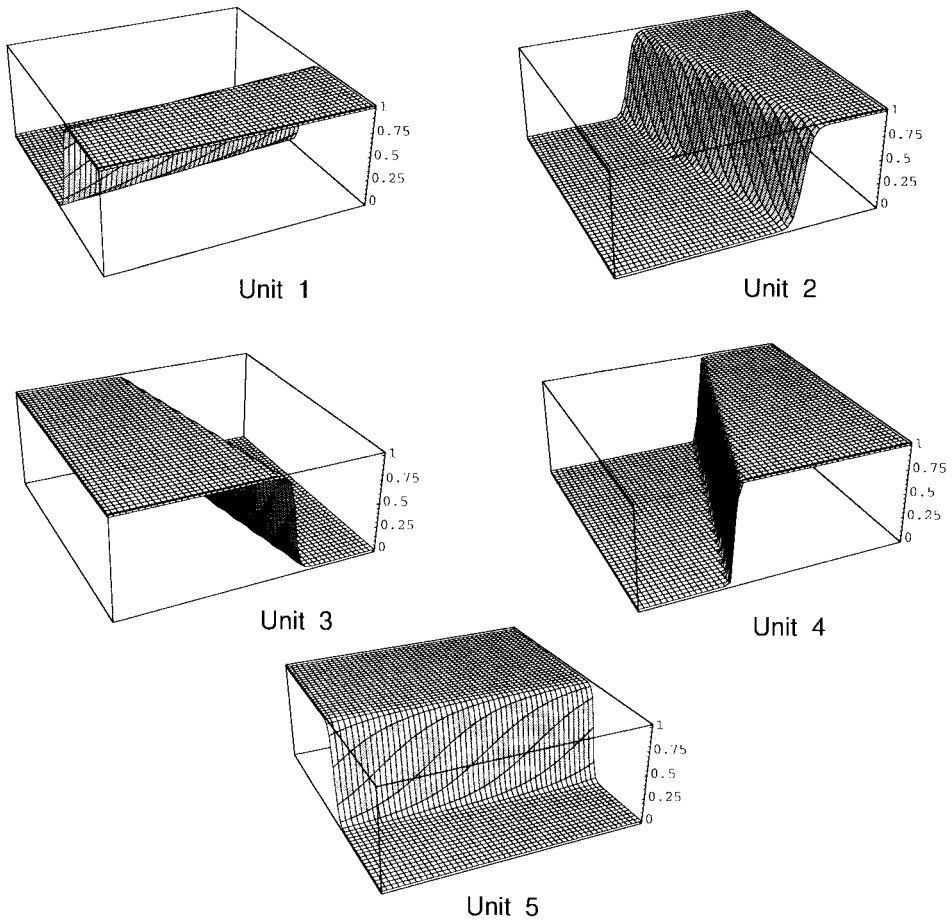



Fig. 9. Responses of the hidden units of the FFNN.

the state 1.0 to the state 0.0 is abrupt and therefore the network crisply discriminates between "circles" and "not circles." This crisp decision-making is consistent with the distribution of the data on the feature space because there is no overlapping of the class "circles" with the other two classes.

Fig. 7(b) shows the decision space generated by the FFNN for the crosses, i.e., the response of the third output unit. The graduated $z$ axis is located at ($-10$,$-10$) on the feature space. Ideally, this unit should respond with values close to 1.0 over $\{x > -2.0\} \cap \{y > 0.0\}$, which is predominantly the region of the feature space over which the crosses are distributed. The response of this unit over the other three quadrants of the feature space should be close to 0.0. Fig. 7(b) does show these properties approximately. However, other properties that are inconsistent with the distribution of the data on the feature space are apparent. There are a few spurious humps in the region of $x < -2.0$. These spurious humps indicate poor generalization and are the result of the raw influence the hidden unit partitions have on the decision surface [24]. It can also be observed from Fig. 7(b) that there is sharp transition in the response from the region of the crosses to the region of the triangles over the plane $x > -2.0$. In addition to abrupt discontinuities of the boundaries, there is a lack of any structure in the overall response and the values of the response are not consistent with the distribution of data points over feature space. Fig. 7(c) shows the decision space generated by the FFNN for the triangles, i.e., the output of the second output unit. The graduated $z$ axis is located at ($-10$, $-10$) on the feature space. Again, neglecting the spurious hump on the $x < -2.0$ plane, all the other properties observed in Fig. 7(b) can be seen preserved in this decision surface also. In particular, it is evident that the transition in the response from the region of the triangles to the region of the crosses in crisp. Moreover, it is also apparent that the structure in the feature space is not reflected in the decision surface. This particular aspect is also explained in detail in the following section.

In the next set of experiments, a QNN with two inputs, five hidden units, and three output units was trained on this data set. In the first set of experiments, the QNN consisted of two-level hidden units. The synaptic weights were updated with a learning rate of $1\times10^{-2}$. The hidden units were tuned with a learning rate of $8\times10^{-2}$. The decision space for the circles generated by the QNN is shown in Fig. 8(a). The graduated $z$ axis is located at (10,$-10$) on the feature space. Fig. 8(a) demonstrates that the QNN is capable of making crisp decisions when the input information is sufficient to make crisp decisions. Since the class of circles does not overlap with the other two classes, the QNN responds crisply for this class, segmenting the feature space into two disjoint subsets, one

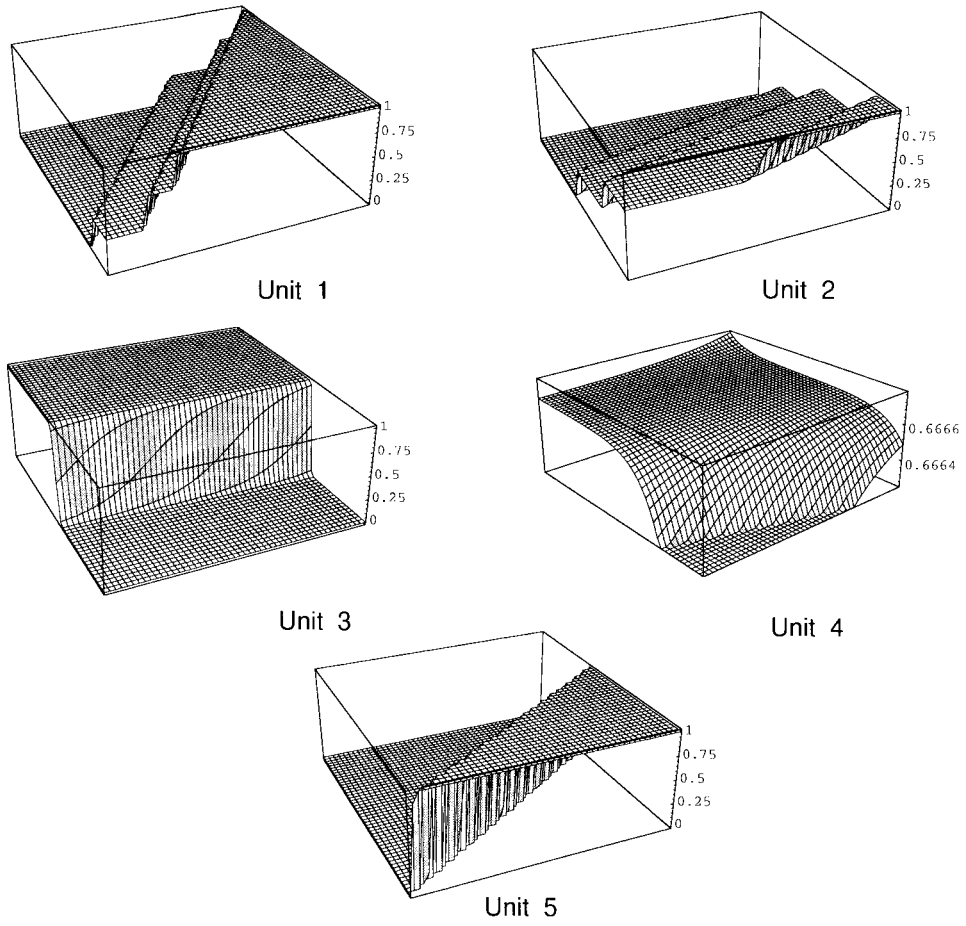



Fig. 10. Responses of the hidden units of the QNN.

representing "circles" and the other "no circles." The quantum interval learning algorithm proposed is functional since the two-level hidden unit transfer functions forming the partition for the circles have been selectively "collapsed-in" to make this partition crisp. These points are further elaborated below with figures showing the hidden unit partitions explicitly.

Fig. 8(b) and (c) show the decision space for the triangles generated by the QNN with two and three-level hidden units, respectively. This fuzzy partition is finer than that seen in Fig. 8(a). Both these responses are very consistent with the manner in which data are distributed on the feature space. In Figs. 8(b) and (c), the response falls off sharply toward the region of the circles and gradually, in graded steps, toward the region of the crosses. At the extremes of the plane $x > -2.0$, there is certainty and this gradually transforms into maximum uncertainty at the boundary between the crosses and the triangles.

Consider the three-class hybrid data set shown in Fig. 6. Fig. 9 shows the responses of all the five hidden units of the FFNN trained over this data set. Fig. 10 shows the responses of all the five hidden units of the corresponding QNN. It is clear from these figures that the FFNN hidden unit partitions are formed so as to ensure that a linear superposition of these partitions has a value close to 1.0 over part of the feature space and a value close to 0.0 over the rest of the feature space. Therefore these representations lack structure. The responses of the hidden units of the QNN have the structure of the feature space reflected in them. These are indeed elemental fuzzy partitions. The ideal decision space of the QNN is a limiting sum of these elemental fuzzy partitions of the feature space. Finally, tuning the hidden representations directly gives an additional degree of freedom for reducing the representation error possibly through a more efficient utilization of the hidden units. This is clearly visible in the decision spaces of the FFNN and the QNN.

### *C. A Two-Class Nonconvex Data Set*

Consider the data set with two nearly concentric classes of overlapping data shown in Fig. 11. This data set has a more complicated structure in the feature space than the previous data sets. Consider the boundary between these two classes to be an annular ring enclosing most of the circles. The density of the crosses around the periphery of this annular ring is not uniform. The boundary between the two classes in the upper and the right regions of the feature space is fairly crisp due to the fact that there is high density of crosses just outside and a high density of circles just inside the boundary. However, the region immediately outside of the boundary at the extreme bottom left is devoid of any crosses. This gives rise to uncertainty at the bottom left segment of the boundary.

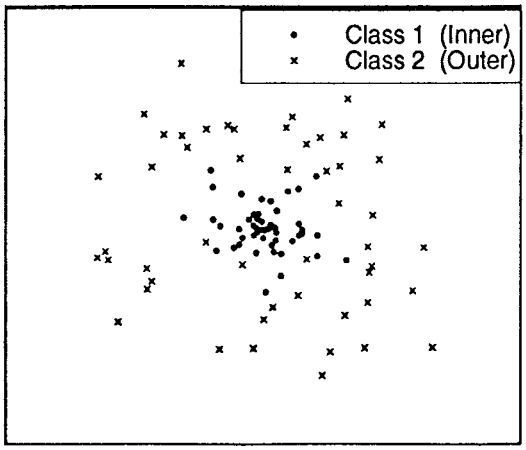
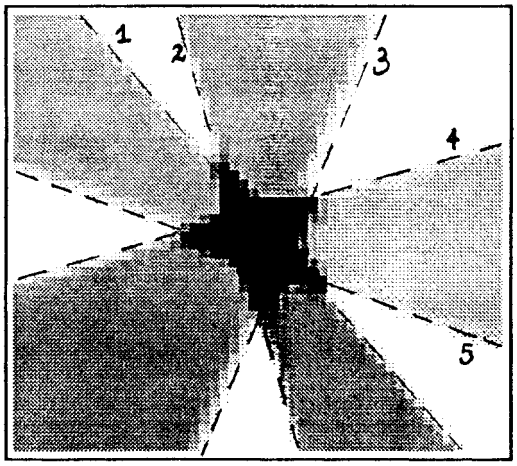
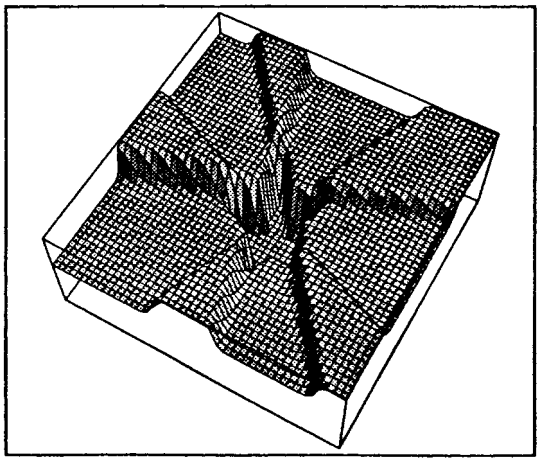



Fig. 11. A two-class nonconvex data set.




(a)



(b)

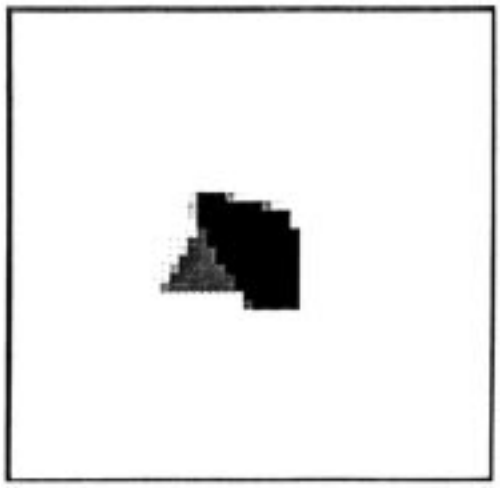
Fig. 12. Response of the FFNN: (a) Density plot and (b) 3-D plot.

In the first experiment, an FFNN with two inputs and two output units was trained to consistently partition the feature space of this data set using the EBP with a learning rate of $1 \times 10^{-2}$. The number of hidden units were varied from three to eight. The FFNN with five hidden units was found to be satisfactory in terms of the minimum error achievable for a given number of adaptation cycles, and the overall time required for training the network. Fig. 12(a) shows the density plot of the first output unit. The bright regions represent response values close to 1.0 and the dark regions represent response values close to 0.0. Fig. 12(b) shows the 3-D plot of the same. Fig. 12(a) and (b) clearly show that the variations in the values of the output, i.e., response values "between zero and one," are irrelevant and do not correspond in any way to the distribution of the data points in the feature space. Consider the five radially outgoing humps in the decision space. As pointed out in the previous section, these humps are spurious and false variations in the network response resulting from the raw influence of the hidden unit partitions on the structure of the decision surface. The decision surface needs to be more or less circular. This circular surface has to be formed as a superposition of five elemental linear partitions. Therefore, the hidden unit partitions are organized in the manner Fig. 12(a) indicates. Thus the variations in the output space only reflect the linear partitions of the hidden units and not the distribution of the data in the feature space. Decreasing the number of hidden units to three made the learning task more difficult and it was observed that the number of misclassifications was seven even after 100 000 cycles of training. Increasing the number of hidden units up to 12 was found to have no considerable "smoothing" effect on the decision surface.

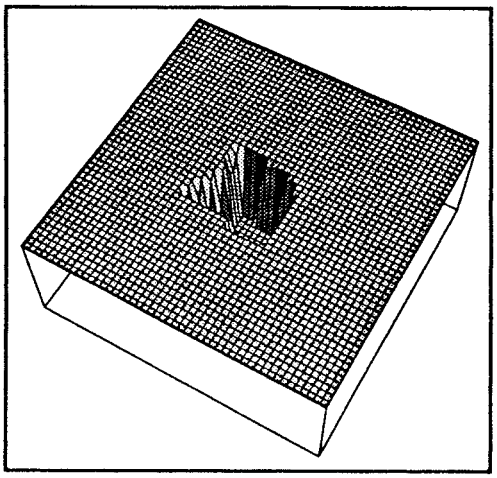
In the next experiment, a QNN with two inputs, two output units, and five hidden units was trained with the same data set. Each hidden unit transfer function was formed with two sigmoids. The decision space is shown in Fig. 13(a) and (b). The intensity variations correspond well with the distribution of the data in the feature space. There are three levels of certainty corresponding to the the two sigmoids in the hidden unit transfer functions. The network response over the region densely distributed with the crosses is close to 1.0, the response over the central region densely distributed with the circles is close to 0.0 and the region to the bottom left corner of the feature space that does not contain any crosses or circles has an intermediate value close to 0.5.

## VI. Conclusions

This paper introduced a new class of feedforward neural networks called QNN's that are capable of learning the uncertainty in sample data. The main aim of the proposed method is to obtain an approximate classification for uncertain data, without any restricting assumptions such as the availability of *a priori* information, limited number of classes of data, limited number of levels of uncertainty in the data, convexity of the classes, etc. In other words, the proposed method does not assign feature vectors that lie in the boundary region between overlapping classes of data exclusively to one particular class only. Instead, such a feature vector is assigned partially to all the classes whose overlapping boundaries include this feature vector. Such an approximate decision is often an end in itself because it is a necessary step in a hierarchical context-based decision-making system. For example, in many



(a)



(b)

Fig. 13. Response of the QNN: (a) density plot and (b) 3-D plot.

character recognition systems, the first stage of the decision making hierarchy is to decide approximately what each letter of a word may be, based only on its shape. In the next stage, based on similar decisions for all adjacent letters in the word, and semantic correctness, the final decisions are made for all the letters of the word [7].

The experiments with simple one-dimensional and two-dimensional data sets showed that conventional FFNN's are often not capable of generalizing the information available from samples consistently (and adequately) to function as fuzzy classifiers. The sensitive dependence of the generalization ability of conventional FFNN's on the topology of the feature space was also demonstrated. This verifies the theory of FFNN classifiers presented in [24]. The experiments also clearly establish the merit of the QNN as an architecture with the ability for recognizing structures in data and organization in the feature space. It is also apparent from the experiments that the manner in which QNN's perceive the structure of a feature space is qualitatively analogous to the manner in which human intuition and judgment assess the physical object space represented by this feature space. The ability of QNN's to adaptively learn the uncertainty in the feature space comes at the cost of an increase in the number of the parameters that need to be learned. Specifically, for each hidden unit there is an additional $n_s$ number of parameters to be updated during each training epoch. On the average, in all the experiments described in the previous section, the computational overhead in terms of overall training time for these additional parameters was about 50%.

Theoretical studies have indicated that this inadequacy of conventional FFNN's is due to the hidden unit functions having just one degree of freedom (their slope) to accommodate for the uncertainty in the feature space [23], [24]. It is therefore expected that radial basis functions are probably a better substitute for sigmoidal functions to achieve this particular function. Further research can study radial basis function networks and compare their performance to the QNN performance.

## Appendix A
## Derivation of the Update Equations for Synaptic Weights

The update equation for the synaptic weight $w_{pj}$ connecting the $p$th output unit to the $j$th hidden unit can be obtained by computing the derivative of $E_k$ with respect to $w_{pj}$ as

$$w_{pj,k} - w_{pj,k-1} = -\alpha \frac{\partial E_k}{\partial w_{pj}} = \alpha \sum_{i=1}^{n_o} (y_{i,k} - \hat{y}_{i,k}) \frac{\partial \hat{y}_{i,k}}{\partial w_{pj}} \tag{A1}$$

where $w_{pj,k-1}$ and $w_{pj,k}$ are the values of $w_{pj}$ before and after the adaptation for the $k$th input and $\alpha$ is the learning rate. The derivative of $\hat{y}_{i,k}$ with respect to $w_{pj}$ can be computed using the definition of $\hat{y}_{i,k}$ in (6) as

$$\frac{\partial \hat{y}_{i,k}}{\partial w_{pj}} = \hat{y}'_{i,k}\, \tilde{h}_{j,k}\, \delta_{ip} \tag{A2}$$

where

$$\hat{y}'_{i,k} = \begin{cases} \beta_o\, \hat{y}_{i,k}(1 - \hat{y}_{i,k}) & \text{if } \hat{y}_{i,k} = \mathrm{sgm}(\beta_o(\overline{y}_{i,k})) \\ 1 & \text{if } \hat{y}_{i,k} = \overline{y}_{i,k}. \end{cases} \tag{A3}$$

and $\delta_{ip} = 1$ if $i = p$ and $\delta_{ip} = 0$ if $i \neq p$. Substituting the above equation into (A1), the update equation is obtained as

$$w_{pj,k} - w_{pj,k-1} = \alpha \epsilon^o_{p,k} \tilde{h}_{j,k} \tag{A4}$$

where

$$\begin{aligned} &\epsilon^o_{p,k} \\ &= (y_{p,k} - \hat{y}_{p,k})\, \hat{y}'_{p,k} \\ &= \begin{cases} \beta_o \hat{y}_{p,k}(1 |!- |!\hat{y}_{p,k})(y_{p,k} - \hat{y}_{p,k}) & \text{if } \hat{y}_{p,k} = \mathrm{sgm}(\beta_o(\overline{y}_{p,k})) \\ (y_{p,k} - \hat{y}_{p,k}) & \text{if } \hat{y}_{p,k} = \overline{y}_{p,k}. \end{cases} \end{aligned} \tag{A5}$$

The update equation for the synaptic weight $v_{sq}$ connecting the $s$th hidden unit to the $q$th input can be obtained by computing the derivative of $E_k$ with respect to $v_{sq}$ as

$$v_{sq,k} - v_{sq,k-1} = -\alpha \frac{\partial E_k}{\partial v_{sq}} = \alpha \sum_{i=1}^{n_o} (y_{i,k} - \hat{y}_{i,k}) \frac{\partial \hat{y}_{i,k}}{\partial v_{sq}} \tag{A6}$$

where $v_{sq,k-1}$ and $v_{sq,k}$ are the values of $v_{sq}$ before and after the adaptation and $\alpha$ is the learning rate. The definition of $\hat{y}_{i,k}$ in (6) gives

$$\frac{\partial \hat{y}_{i,k}}{\partial v_{sq}} = \hat{y}'_{i,k} \sum_{j=0}^{n_h} w_{ij} \frac{\partial \tilde{h}_{j,k}}{\partial v_{sq}} \tag{A7}$$

with $\hat{y}'_{i,k}$ as defined in (A3). Finally, the definition of $\tilde{h}_{j,k}$ in (5) gives

$$\frac{\partial \tilde{h}_{j,k}}{\partial v_{sq}} = \beta_h \frac{1}{n_s} \sum_{r=1}^{n_s} h^r_{j,k} (1 - h^r_{j,k}) \delta_{js} x_{q,k}. \tag{A8}$$

Substituting (A7) and (A8) in (A6), the update equation for $v_{sq}$ becomes

$$v_{sq,k} - v_{sq,k-1} = \alpha \beta_h \epsilon^h_{s,k} x_{q,k} \tag{A9}$$

where

$$\epsilon^h_{s,k} = \left( \frac{1}{n_s} \sum_{r=1}^{n_s} h^r_{s,k} (1 - h^r_{s,k}) \right) \sum_{i=1}^{n_o} \epsilon^o_{i,k} w_{is}. \tag{A10}$$

## Appendix B
## Derivation of the Update Equations for Quantum Intervals

The update equation for $\theta^s_q$ can be obtained by setting the change in $\theta^s_q$, say $\Delta\theta^s_q$, proportional to the gradient of $G$ with respect to $\theta^s_q$ as

$$\begin{aligned} \Delta\theta^s_q &= -\alpha_\theta \frac{\partial G}{\partial \theta^s_q} \\ &= -\alpha_\theta \sum_{m=1}^{n_o} \sum_{\mathbf{x}_k : \mathbf{x}_k \in \mathcal{C}_m} (\langle \tilde{h}_{q,\mathcal{C}_m} \rangle - \tilde{h}_{q,k}) \\ &\quad \times \left( \frac{\partial \langle \tilde{h}_{q,\mathcal{C}_m} \rangle}{\partial \theta^s_q} - \frac{\partial \tilde{h}_{q,k}}{\partial \theta^s_q} \right) \end{aligned} \tag{B1}$$

where $\alpha_\theta$ is the learning rate. The definition of $\langle \tilde{h}_{q,\mathcal{C}_m} \rangle$ in (12) gives

$$\frac{\partial \langle \tilde{h}_{q,\mathcal{C}_m} \rangle}{\partial \theta^s_q} = \frac{1}{|\mathcal{C}_m|} \sum_{\mathbf{x}_k : \mathbf{x}_k \in \mathcal{C}_m} \frac{\partial \tilde{h}_{q,k}}{\partial \theta^s_q}. \tag{B2}$$

The definition of $\tilde{h}_{q,k}$ in (5) gives

$$\begin{aligned} \frac{\partial \tilde{h}_{q,k}}{\partial \theta^s_q} &= \frac{1}{n_s} \sum_{r=1}^{n_s} \frac{\partial h^r_{q,k}}{\partial \theta^s_q} \\ &= -\beta_h \frac{1}{n_s} \sum_{r=1}^{n_s} h^r_{q,k} (1 - h^r_{q,k}) \delta_{rs} = -\frac{\beta_h}{n_s} \nu^s_{q,k} \end{aligned} \tag{B3}$$

where $\nu^s_{q,k} = h^s_{q,k}(1 - h^s_{q,k})$. Substituting (B2) and (B3) in (B1) gives the update equation as

$$\begin{aligned} &\Delta\theta^s_q \\ &= \alpha_\theta \frac{\beta_h}{n_s} \sum_{m=1}^{n_o} \sum_{\mathbf{x}_k : \mathbf{x}_k \in \mathcal{C}_m} (\langle \tilde{h}_{q,\mathcal{C}_m} \rangle - \tilde{h}_{q,k}) (\langle \nu^s_{q,\mathcal{C}_m} \rangle - \nu^s_{q,k}) \end{aligned} \tag{B4}$$

where $\langle \nu^s_{q,\mathcal{C}_m} \rangle = \frac{1}{|\mathcal{C}_m|} \sum_{\mathbf{x}_k : \mathbf{x}_k \in \mathcal{C}_m} \nu^s_{q,k}$.

## References


[1] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[2] J. C. Bezdek and S. K. Pal, Eds., *Fuzzy Models for Pattern Recognition: Models that Search for Structures in Data*. New York: IEEE Press, 1992.

[3] V. Cherkassky, J. H. Friedman, and H. Wechsler, Eds., *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*. Berlin: Springer-Verlag, 1994.

[4] G. Cybenko, "Approximations by superposition of sigmoidal function," *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.

[5] A. DeLuca and S. Termini, "Entropy and energy measures of a fuzzy set," in *Advances in Fuzzy Set Theory and Applications*, M. M. Gupta, R. K. Ragade, and R. R. Yager, Eds. Amsterdam, The Netherlands: North-Holland, 1979, pp. 321–338.

[6] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[7] D. Hammerstrom, "A digital VLSI architecture for real-world applications," in *An Introduction to Neural and Electronic Networks*, S. F. Zornetzer, J. L. Davis, C. Lau, and T. McKenna, Eds. New York: Academic, 1995.

[8] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 801–806, 1992.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[10] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Possibility and necessity pattern classification using neural networks," *Fuzzy Sets Syst.*, vol. 48, pp. 331–340, 1992.

[11] H. Ishibuchi and H. Tanaka, "Approximate pattern classification using neural networks," in *Fuzzy Logic: State of the Art*, R. Lowern and M. Roubens, Eds. Dordrecht, The Netherlands: Kluwer, 1993.

[12] Y. Ito, "Extension of approximation capability of three layered neural networks to derivatives," in *Proc. IEEE Int. Conf. Neural Networks*, 1993, pp. 653–658.

[13] J. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, 1993.

[14] A. Kandel, *Fuzzy Expert Syst.* Boca Raton, FL: CRC, 1992.

[15] N. B. Karayiannis and A. N. Venetsanopoulos, *Artificial Neural Networks: Learning Algorithms, Performance Evaluation and Applications*. Boston, MA: Kluwer, 1993.

[16] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 7, pp. 693–699, 1985.

[17] J. M. Keller, R. R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic," *Fuzzy Sets Syst.*, vol. 45, pp. 1–12, 1992.

[18] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[19] S. K. Pal and D. K. Dutta Majumder, Fuzzy *Mathematical Approach to Pattern Recognition*. New York: Wiley, 1986.

[20] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–697, 1992.

[21] G. Purushothaman, "Fuzzy neural processing for pattern recognition," Master's thesis, Detp. Electr. Computer Eng., Univ. Houston, TX, Dec. 1994.
[22] G. Purushothaman and N. B. Karayiannis, "Feedforward neural architectures for membership estimation and fuzzy classification," in *Intelligent Engineering Systems Through Artificial Neural Networks*, C. H. Dagli *et al*., Eds. New York: ASME Press, 1994, pp. 235–240.
[23] ———, "On the capacity of feedforward neural networks for fuzzy classification," in *Intelligent Engineering Systems Through Artificial Neural Networks*, C. H. Dagli *et al*., Eds. New York: ASME Press, 1995, pp. 253–258.
[24] G. Purushothaman and N. B. Karayiannis, "On the capacity of feedforward neural networks for fuzzy classification," submitted to *IEEE Trans. Syst., Man. Cybern.*
[25] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. Neural Networks*, vol. 1, pp. 296–298, 1990.
[26] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986.
[27] M. Sugeno, *Industrial Applications of Fuzzy Control*. New York: Elsevier, 1985.
[28] H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning," *Int. J. Approximate Reasoning*, vol. 5, no. 3, pp. 191–212, 1991.
[29] T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Its Applications*. New York: Academic, 1991.
[30] L. A. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338–353, 1965.
[31] ———, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. 3, pp. 28–44, 1973.
[32] L. A. Zadeh, K. S. Fu, K. Tanaka, and M. Shimura, Eds., *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*. New York: Academic, 1975.

**Gopathy Purushothaman** received the B.E. degree from the University of Madras, India, in 1989, the M.Tech. degree from the Indian Institute of Technology, Kanpur, in 1990, and the M.S. degree from the University of Houston, TX, in 1994, in electrical and computer engineering. He is presently a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of Houston.

Since 1993, he has been a Research and Teaching Assistant in the Department of Electrical and Computer Engineering at the University of Houston. From 1991 to 1992 he was a Lecturer in Electrical and Computer Engineering at the College of Engineering, Pondicherry, India. His research interests include neural networks, psychophysics and neurodynamics of vision, and applications of neural networks and fuzzy logic.

Mr. Purushothaman is a corecipient of a Theoretical Development Award for a paper presented at the Artificial Neural Networks in Engineering '94 Conference. He also received the Urvish Medh Memorial Award from the Department of Electrical and Computer Engineering, University of Houston, in 1995, the Jawaharlal Nehru Memorial Award from the Government of India in 1990, and the University of Madras Gold Medal in 1989. He is a Life Member of the Indian Society for Technical Education.

**Nicolaos B. Karayiannis** (S'85-M'91), for a photograph and biography, see this issue, p. 518.