Fuzzy-Rough Neural Networks for Vowel Classification

Manish Sarkar and B. Yegnanarayana
Department of Computer Science & Engineering
Indian Institute of Technology, Madras - 600 036, INDIA
E-mail: manish@bronto.iitm.ernet.in

ABSTRACT

While designing radial basis function neural networks for classification, fuzzy clustering is often used to position the hidden nodes in the input space. The main assumption of the clustering is that similar inputs produce similar outputs. In other words, it means that any two input patterns from the same cluster must be from the same class. Generalization is possible in the radial basis function neural networks due to this similarity property. In many real life applications, however, two patterns from the same cluster belong to different classes, and hence, classification based on mere similarity property is inadequate. This problem arises because the available features are not sufficient to discriminate the classes. It implies that the fuzzy clusters generated by the input features have rough uncertainty. This paper proposes a fuzzy-rough set based network which exploits fuzzy-rough membership functions to reduce this problem. The proposed network is theoretically a powerful classifier as it is equivalent to a universal approximator. Moreover, its activity is transparent as it can easily be mapped to a Takagi-Sugeno type fuzzy rule base system. The efficacy of the proposed method is studied on a vowel recognition problem.

Keywords: Fuzzy sets, rough sets, fuzzy-rough sets, clustering, fuzzy-rough membership functions, radial basis function networks, Takagi-Sugeno type fuzzy rule base systems.

I. INTRODUCTION

Among various kinds of neural networks, feedforward neural networks, based on the backpropagation learning algorithm [5], are extensively used for pattern classification. However, a major drawback of the backpropagation algorithm is that it may not ensure successful learning because of local minima problems and a long training time. A radial basis function neural network (RBFNN) [5], a class of three layered feedforward neural networks, has faster local minima free learning capability than the other feedforward neural networks with backpropagation learning algorithm. However, the RBFNN employs one hidden node completely for each input training pattern, and hence, it suffers from a huge memory overhead while learning a large number of training examples. Moreover, the presence of a large number of hidden nodes in turn increases the time required to classify each test pattern. Hence, supervised or unsupervised fuzzy clustering of the input data set is used to put a bigger Gaussian at each hidden node, which subsequently reduces the number of hidden nodes. The main idea of using fuzzy clustering is that if two input patterns are similar, i.e., close neighbors in the input pattern space, then the class labels associated with them will be same. When a new pattern is presented at the input layer of the network, the network classifies precisely based on this similarity or neighborhood property. Thus, the inherent similarity or neighbourhood property of the clusters leads the network to generalize. Since each cluster in the pattern space represents certain common property, it is logical that the patterns from the same cluster will also belong to the same class. In real life cases, however, we cannot extract all the relevant features necessary for the classification. Consequently, two patterns may have similar features, but they are not similar if the other features, including the existing ones, are accounted for. Therefore, when the input patterns are clustered based on the available features, two apparently similar or neighboring patterns may have different class labels. Thus, the clustering techniques are not helpful to reduce the number of hidden nodes as the same cluster may have more than one pattern belonging to different classes. It makes the output classes *indiscernible* or indistinguishable based on the given set of features. For example, if the objects are classified according to color (red, black) and shape (triangle, square and circle), then the indiscernible classes are: red triangles, black squares, red circles, etc. Thus, these two attributes make a *partition* or forms clusters in the set of objects and the universe becomes *coarse*. Now, if two red triangles with different areas belong to different classes, it is impossible for anyone to classify these two red triangles based on the given two attributes. This kind of uncertainty is referred to as *rough uncertainty* [11]. One way to completely avoid the rough uncertainity is to extract the essential features so that distinct feature vectors are used to represent different objects. But, it may not be possible to guarantee this as our knowledge about the system generating the data is limited. Another way to avoid rough uncertainity is to break the clusters further so that they do not contain any pattern from the other classes. This is difficult as each fuzzy cluster overlaps patterns from the other classes to some extent. Moreover, the breaking of clusters means destruction of the similarity property, which in turn means the destruction of the generalization property. In addition, if the clusters are broken too much, then the RBFNN training may need large space and high time complexity.

In this paper, we attempt to reduce the effect of rough uncertainity, while keeping the similarity property intact. To tackle the similarity property we need fuzzy sets [7], and to tackle the roughness we need rough sets [11]. Since both fuzziness and roughness are present here, we incorporate both fuzzy and rough sets on a common platform, called *fuzzy-rough* sets [3]. To manipulate the fuzzy-rough sets, we use *fuzzy-rough membership functions* [16]. The fuzzy-rough membership function is further exploited to construct a *fuzzy-rough neural network* (FRNN). Basically, the FRNN uses the fuzzy linguistic uncertainity involved in the input data set and the roughness present in the input-output relationship. On the otherhand, the RBFNN captures only the fuzzy linguistic uncertainity. Although in absence of the roughness the FRNN behaves like an RBFNN, in presence of the roughness the FRNN performs significantly better than its RBFNN counterpart. One advantage of the classification procedure used in the FRNN is that it is *possibilistic* [7]. Specifically, it is useful when the output of the FRNN is again used for further classification. Theoretically, the FRNN is a powerful classifier as it can be shown to be a universal approximator. Unlike the feedforward neural networks with backpropagation algorithm, the FRNN does not act like a *black box*, but works like a transparent one. In fact, the FRNN can be viewed as a Takagi-Sugeno type fuzzy rule base system [5] [18]. This transparency aids us to have a greater insight about the whole classification process performed by the FRNN. Moreover, the FRNN can be used to extract the fuzzy rules hidden inside the data.

II. BACKGROUND

A. Rough Sets

Let R be an equivalence relation on a universal set X. Moreover, let X/R denote the family of all equivalence classes

This work was carried out while Manish Sarkar held a Dr. K. S. Krishnan Research Fellowship from Department of Atomic Energy, Government of India.

induced on X by R. One such equivalence class in X/R, that contains $x \in X$, is designated by $[x]_R$. For any output class $A \subseteq X$, we can define the lower $\overline{R}(A)$ and upper $\underline{R}(A)$ approximations, which approach A as closely as possible from inside and outside, respectively [7]. Here,

$$\underline{R}(A) = \cup\{[x]_R \mid [x]_R \subseteq A,\ x \in X\} \qquad \text{(1-a)}$$

is the union of all equivalence classes in X/R that are contained in A, and

$$\overline{R}(A) = \cup\{[x]_R \mid [x]_R \cap A \neq \phi,\ x \in X\} \qquad (2)$$

is the union of all equivalence classes in X/R that overlap with A. A rough set $R(A) = \langle \overline{R}(A), \underline{R}(A) \rangle$ is a representation of the given set A by $\underline{R}(A)$ and $\overline{R}(A)$ [12]. The set $BN(A) = \overline{R}(A) - \underline{R}(A)$ is a rough description of the boundary of A by the equivalence classes of X/R. The approximation is rough uncertainty free if $\overline{R}(A) = \underline{R}(A)$. Thus, when all the patterns from an equivalence class do not carry the same output class label, rough ambiguity is generated as a manifestation of the one-to-many relationship between that equivalence class and the output class labels.

The *rough membership function* $r_A(x) : A \rightarrow [0,\ 1]$ of a pattern $x \in X$ for the output class A is defined by [19]

$$r_A(x) = \frac{\|[x]_R \cap A\|}{\|[x]_R\|} \qquad (2)$$

where $\|A\|$ denotes the cardinality of the set A.

B. Fuzzy Sets

In traditional two-state classifiers, where a class A is defined as a subset of a universal set X, any input pattern $x \in X$ can either be a member or not be a member of the given class A. This property of whether or not a pattern x of the universal set belongs to the class A can be defined by a *characteristic function* $\mu_A : X \rightarrow \{0, 1\}$ as follows:

$$\mu_A(x) = \begin{cases} 1 & \text{if and only if} \quad x \in A \\ 0 & \text{if and only if} \quad x \notin A \end{cases}$$

In real life situations, however, boundaries between the classes may be overlapping. Hence, it is uncertain whether an input pattern belongs totally to the class A. To take care of such situations, in fuzzy sets [1] the concept of characteristic function has been modified to *membership function* $\mu_A : X \rightarrow [0,\ 1]$. This function is called membership function, because larger value of the function denotes more membership of the element to the set under consideration.

C. Rough-Fuzzy Sets

Let X be a set, R be an equivalence relation defined on X and the output class $A \subseteq X$ be a fuzzy set. A rough-fuzzy set is a tuple $\langle \overline{R}(A), \underline{R}(A) \rangle$, where the lower approximation $\underline{R}(A)$ and the upper approximation $\overline{R}(A)$ of A are fuzzy sets of X/R, with membership functions defined by [3]

$$\mu_{\underline{R}(A)}([x]_R) = \inf\{\mu_A(x) \mid x \in [x]_R\} \qquad \text{(3-a)}$$
$$\mu_{\overline{R}(A)}([x]_R) = \sup\{\mu_A(x) \mid x \in [x]_R\} \qquad \text{(3-b)}$$

Here, $\mu_{\underline{R}(A)}([x]_R)$ and $\mu_{\overline{R}(A)}([x]_R)$ are the membership values of $[x]_R$ in $\underline{R}(A)$ and $\overline{R}(A)$, respectively.

The rough-fuzzy membership function of a pattern $x \in X$ for the fuzzy output class $C_c \subseteq X$ is defined by [15]

$$\iota_{C_c}(x) = \frac{\|F \cap C_c\|}{\|F\|} \qquad (4)$$

where $F = [x]_R$ and $\|C_c\|$ is equal to the cardinality of the fuzzy set C_c. One possible way to determine the cardinality is to use [20] $\|C_c\| \overset{def}{=} \sum_{x \in X} \mu_{C_c}(x)$. For the '$\cap$' (intersection) operation, we can use [20] $\mu_{A \cap B}(x) \overset{def}{=} \min\{\mu_A(x), \mu_B(x)\}$ $\forall x \in X$. When the output class is crisp, the definition (4) boils down to the definition (2).

D. Fuzzy-Rough Sets

When the equivalence classes are not crisp, they are in form of fuzzy clusters $\{F_1, F_2, \ldots, F_H\}$ generated by a *fuzzy weak partition* [3] of the input set X. The term fuzzy weak partition means that each F_j is a normal fuzzy set, i.e., $\max_x \mu_{F_j}(x) = 1$ and $\inf_x \max_j \mu_{F_j}(x) > 0$ while

$$\sup_x \min\{\mu_{F_i}(x), \mu_{F_j}(x)\} < 1 \quad \forall\, i,\ j \in \{1, 2, \ldots, H\} \qquad (5)$$

Here $\mu_{F_j}(x)$ is the fuzzy membership function of the pattern x in the cluster F_j. In addition, the output classes C_c, $c = \{1, 2, \ldots, C\}$ may be fuzzy too. Given a weak fuzzy partition $\{F_1, F_2, \ldots, F_H\}$ on X, the description of any fuzzy set C_c by means of the fuzzy partitions under the form of an upper and a lower approximation $\overline{C_c}$ and $\underline{C_c}$ is as follows:

$$\mu_{\underline{C_c}}(F_j) = \inf_{x \in C_c} \max\{1 - \mu_{F_j}(x), \mu_{C_c}(x)\}\ \forall\, x \qquad \text{(6-a)}$$
$$\mu_{\overline{C_c}}(F_j) = \sup_{x \in C_c} \min\{\mu_{F_j}(x), \mu_{C_c}(x)\}\ \forall\, x \qquad \text{(6-b)}$$

The tuple $\langle \underline{C_c}, \overline{C_c} \rangle$ is called a fuzzy-rough set. Here, $\mu_{C_c}(x) = \{0,\ 1\}$ is the fuzzy membership of the input x to the class C_c. Fuzzy-roughness appears when a fuzzy cluster contains patterns that belong to different classes.

III. FUZZY-ROUGH MEMBERSHIP FUNCTIONS

A. Definition

If the equivalent classes form the fuzzy clusters $\{F_1, F_2, \ldots, F_H\}$, then each fuzzy cluster can be considered as a fuzzy linguistic variable. Now, we generalize the definition (4) to the following definition of the fuzzy-rough membership function [16]:

$$\tau_{C_c}(x) = \begin{cases} \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} \mu_{F_j}(x) \iota^j_{C_c}(x) & \text{if } \exists j \text{ with } \mu_{F_j}(x) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (7)$$

where $\hat{H}$ is the number of cluster in which x has non zero membership and $\iota^j_{C_c}(x) = \frac{\|F_j \cap C_c\|}{\|F_j\|}$. Here, $\tau_{C_c}(x)$ represents the fuzzy-rough uncertainity of x in the class C_c.

B. Properties

1. $0 \leq \tau_{C_c}(x) \leq 1$
   Proof: Since $\phi \subseteq F_j \cap C_c \subseteq F_j$, $0 \leq \iota^j_{C_c}(x) \leq 1$. Moreover, $0 \leq \mu_{F_j}(x) \leq 1$. Hence, the proof follows.

2. *$\tau_{C_c}(x) = 1$ or 0 if and only if no fuzzy-rough uncertainty is associated with the pattern x.*
   Proof:
   If part: If no fuzzy-rough uncertainty is involved, then x must belong completely to all the clusters in which it has non-zero belongingness. It implies $\mu_{F_j}(x) = 1$ for which $\mu_{F_j}(x) > 0$. Moreover, all the clusters in which x has non-zero belongingness either (a) must be the subsets of the class C_c, or (b) must not share any

pattern with the class C_c. In other words, the condition (a) implies that $F_j \subseteq C_c \; \forall \, j$ for which $\mu_{F_j}(x) > 0$. Hence, $\tau^p_{C_c}(x) = \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} 1.1 = 1$. Similarly the condition (b) expresses that $F_j \cap C_c = \phi \; \forall \; j$ for which $\mu_{F_j}(x) > 0$. Hence, $\tau^p_{C_c}(x) = \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} \mu_{C_c}(x).0 = 0$

Only if part: If $\tau^p_{C_c}(x) = 0$, then x may not belong to any cluster. Therefore, in this case there is no fuzzy-roughness associated with x. Otherwise, each term under the summation symbol, i.e., $\mu_{F_j}(x)\iota^j_{C_c}(x)$ is separately zero. It implies that either $\mu_{F_j}(x)$ or $\iota^j_{C_c}(x)$, or both $\mu_{F_j}(x)$ and $\iota^j_{C_c}(x)$ are zero. If $\mu_{F_j}(x) = 0$, then the pattern x does not belong to the cluster F_j, and hence, no fuzzy-rough uncertainty is involved with x. If $\iota^j_{C_c}(x) = 0$, then F_j and C_c do not have any pattern common, and therefore, no fuzzy-rough uncertainty exists with x. Thus, $\tau^p_{C_c} = 0$ implies that fuzzy-roughness is not associated with the pattern x. If $\tau^p_{C_c} = 1$, then $\mu_{F_j} = 1$ and $\iota^j_{C_c} = 1, \;\; \forall j = 1, 2, \ldots, \hat{H}$. It also indicates the absence of fuzzy-roughness.

It is to be noted here that if $\hat{H} > 1$, $\tau^p_{C_c} \neq 0$ and fuzzy-rough uncertainty is absent, then $\tau^p_{C_c}$ never becomes one, rather it approaches towards one. It is because, the condition expressed in (5) does not allow $\mu_{F_j}(x) = 1$ to be true for more than one cluster. However, it hardly happens in practice as it needs two cluster centers to be same.

3. *If no fuzzy linguistic and fuzzy classification uncertainties are associated with the pattern x, then $\tau_{C_c}(x) = r_{C_c}(x)$.*
Proof: If no fuzzy linguistic uncertainty is involved, then each cluster is crisp. Consequently, the input pattern belongs to only one cluster. Let it be the jth cluster. So, $\mu_{F_j}(x) = 1$ and $\mu_{F_k}(x) = 0 \; \forall k \neq j$. Since the classification is crisp, from the definition (2), $\tau_{C_c}(x) = \frac{\|F_j \cap C_c\|}{\|F_j\|} = r_{C_c}(x)$.

4. *When each cluster is crisp and fine, that is, each cluster consists of a single pattern and the associated cluster memberships are crisp, $\tau_{C_c}(x)$ is equivalent to the fuzzy membership of x in the class C_c. If the output class is also crisp, then $\tau_{C_c}(x)$ is equivalent to the characteristic function value of x in the class C_c.*
Proof: Since each cluster is crisp and *fine*, $\tau_{C_c}(x) = 1.\frac{\mu_{C_c}(x)}{1} = \mu_{C_c}(x)$. In addition, if the output class is crisp, then $\tau_{C_c}(x)$ lies in $\{0, 1\}$, and thus, it becomes the characteristic function.

5. *For a C-class classification problem with crisp output classes, the fuzzy-rough membership functions behave in a possibilistic manner.*
Proof:

$$\sum_{c=1}^{C} \tau^p_{C_c}(x) = \frac{1}{\hat{H}} \sum_{c=1}^{C} \sum_{j=1}^{\hat{H}} \mu_{F_j}(x) \frac{\|F_j \cap C_c\|}{\|F_j\|}$$

$$= \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} \mu_{F_j}(x) \frac{\sum_{c=1}^{C} \sum_y \min\{\mu_{F_j}(y), \mu_{C_c}(y)\}}{\sum_y \mu_{F_j}(y)}$$

$$= \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} \mu_{F_j}(x) \frac{\sum_{c=1}^{C} \sum_{y \in C_c} \min\{\mu_{F_j}(y)\}}{\sum_y \mu_{F_j}(y)}$$

$$= \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} \mu_{F_j}(x) \frac{\sum_y \mu_{F_j}(y)}{\sum_y \mu_{F_j}(y)}$$

$$= \frac{1}{\hat{H}} \sum_{j=1}^{\hat{H}} \mu_{F_j}(x) \qquad (8)$$

Since $\sum_{c=1}^{C} \tau^p_{C_c}(x)$ needs not to be equal to a constant, the resultant classification procedure is possibilistic [7] [9]. Hence, $\tau^p_{C_c}(x)$ can distinguish between equal evidence and ignorance, which are well discussed in belief theory [17] and possibility theory [7].

IV. FUZZY-ROUGH NEURAL NETWORKS

A. Architecture

The FRNN is designed such that it works as a fuzzy-rough membership function, i.e., the outputs of the networks are fuzzy rough membership values corresponding to the input. The proposed FRNN is a three layered feedforward network with one hidden layer (Fig. 1). The number of nodes in the input, hidden and output layers are equal to the dimension of the input pattern (=N), number of the fuzzy clusters present in the input data (=H) and number of the classes (=C), respectively. When an input pattern $\mathbf{x} = (x_1, x_2, \ldots, x_N)'$ is applied at the input layer of the network, the output of the jth hidden node is

$$o^h_j = \exp\left[-\frac{(\mathbf{x} - \mathbf{m}_j)'(\mathbf{x} - \mathbf{m}_j)}{2\sigma_j^2}\right] \qquad (9)$$

where $\mathbf{m}_j$ and σ_j are the center and spread of the Gaussian function used in the jth hidden node, respectively. The center and spread of the hidden nodes can be determined by making them equal to the mean and variance of each cluster. Therefore, the parameters necessary for the FRNN can be obtained from the parameters defined in the input-output space (Table I). The output of the kth output node is

$$o^o_k = \sum_{j=1}^{H} o^h_j w_{jk} \qquad (10)$$

where w_{jk} is the weight from the jth hidden node to the kth output node. The output value o^o_k lies in between 0 and 1 (from property 1) as the output is the fuzzy-rough membership value corresponding to the input. Moreover, from the property 5, o^o_k is possibilistic.

To design the FRNN, the last task is to adjust the weights between the hidden layer and the output layer through training. Precisely, the outputs of the hidden nodes are the fuzzy linguistic membership values (equation (9)), and the weights between the hidden and the output layer reflect the rough-fuzzy membership values. The use of rough-fuzzy membership functions makes the FRNN more powerful than its RBFNN counterpart.

B. Training and Testing

For training, all the weights, $w_{jk}(0) \; \forall j, k$, are initialised to zero. For each input training pattern, the weight adjustment in the first iteration is carried out as

$$\Delta w_{jk}(1) = o^h_j * t \quad \forall j, \, k \qquad (11)$$

where $t = 1$ if $\mathbf{x} \in C_k$ else $t = 0$. It is interesting to note that the training process takes exactly one iteration to be over. After the whole cycle is over, w_{jk} represents $\|F_j \cap C_k\|$, i.e., $\sum_{x \in C_k} \mu_{F_j}(x)$. To make $w_{jk} = \iota^j_{C_k}(x)$, w_{jk} is normalized as $\frac{w_{jk}}{\sum_k w_{jk}}$ (since $\|F_j\| = \sum_k \sum_{x \in C_k} \mu_{F_j}(x) = \sum_k w_{jk}$). Since all the hidden nodes are using Gaussian clusters, each input

TABLE I
THE RELATIONSHIP BETWEEN THE PARAMETERS USED IN FUZZY-ROUGH NEURAL NETWORKS AND INPUT-OUTPUT SPACE

Fuzzy-Rough neural networks		Input-Output Space
No. of the input nodes	=	Dimension of the inputs
No. of the hidden nodes	=	No. of the clusters
No. of the output nodes	=	No. of the classes
Center of jth hidden node	=	Center of the jth cluster
Width of the jth hidden node	=	Width of the jth cluster

pattern belongs to all the clusters, and hence, $\hat{H} = H$. Finally, the weights are set as $w_{jk} = \frac{w_{jk}}{\hat{H}}$ to take care of the term $\hat{H}$ involved in (7). It is to be noted that no bias term is involved here with any node.

In the testing stage, a separate set of test patterns is given as the inputs to the network. For the test input $\mathbf{x}$, the generated output at the cth output node is the fuzzy-rough membership value $\tau_{C_c}(\mathbf{x})$. The output node with the maximum output value indicates the class label of $\mathbf{x}$.

C. Implementation Details

Determination of the fuzzy-rough membership values depend on the fuzzy clustering of the input data set. The fuzziness associated with the clusters represent the fuzzy linguistic uncertainty present in the input data set. The clustering can be performed by

1. *Unsupervised Clustering*: It involves taking the data from all the classes and cluster them subsequently without considering the associated class labels with the data.
2. *Supervised Clustering*: Separate data sets are formed for each class, and clustering is performed on each such data set to find the subgroups present in the input data from the same class.

Both the clustering can be done by fuzzy K-means clustering algorithm [1]. But, the problems with it are: a) the number of clusters has to be fixed *a priori*, which may not be known, b) it will not work if the number of clusters is one, and c) generated fuzzy memberships are not possibilistic. To overcome the first problem, the evolutionary programming-based method, used in [14], can be used. In many cases, this method can automatically determine the number of clusters. Other cluster validity measures [2] can also be used along with the fuzzy K-means algorithm to determine the number of clusters. For the second problem, if we know that only one cluster is present for a class, then we can find the mean and standard deviation from the input data set, and we can fit a π fuzzy membership model [10]. To overcome the third problem, the possibilistic K-means clustering algorithm [8] can be used.

When the input clusters are crisp and the output classes are crisp, the outputs of the network are rough membership values (property 3). Hence, the resultant FRNN architecture is reduced to a modified architecture, called *rough neural networks* (RNN). The architectural difference between the FRNN and the RNN is in the transfer function used in each hidden node. In particular, the transfer function used in the FRNN is of Gaussian type, whereas in the case of RNN the transfer function is a unit gate function, i.e.,

$$o_j^h = \begin{cases} 1 & \text{if} \quad (\mathbf{x}-\mathbf{m}_j)'(\mathbf{x}-\mathbf{m}_j) \le 2\sigma_j^2 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Evidently, the 1/0 option used in the gate function makes the generalization capability of the RNN limited.

When the input clusters are crisp and *fine*, and the output classes are crisp, then from the property 4, the outputs of the FRNN are the crisp class membership values. The resultant network can be called *crisp neural network* (CNN). The number of hidden nodes of the CNN is equal to the number of inputs. Since the width of each cluster approaches towards zero, the transfer function of each hidden node becomes a unit impulse function, i.e.,

$$o_j^h = \begin{cases} 1 & \text{if} \quad \mathbf{x} = \mathbf{x}_j \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

As a result, the weight calculation becomes very simple, i.e., w_{jk} =1 if $\mathbf{x}_j \in C_k$ else $w_{jk} = 0$. Thus, the resultant CNN needs a large amount of space, and it works like a look-up table, which does not have any generalization capability, but has a very good memorising power.

For further discussions, we will assume the structure of the FRNN as a general one, i.e., the one whose architecture is described in the section iv-A.

D. Universal Approximation

It is easy to notice that architecturally (although functionally not) the FRNN is equivalent to the RBFNN. Since the RBFNN is a universal approximator [5], the FRNN is also a universal approximator.

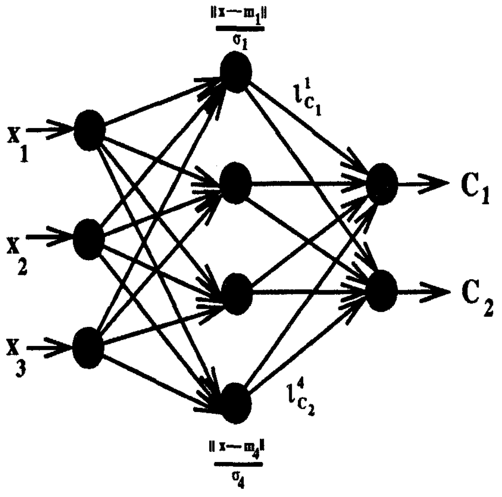
$\frac{\|\mathbf{x}-\mathbf{m}_1\|}{\sigma_1}$
$l^1_{C_1}$
X_1
X_2
X_3
C_1
C_2
$l^4_{C_2}$
$\frac{\|\mathbf{x}-\mathbf{m}_4\|}{\sigma_4}$

Fig. 1. A typical FRNN with three input nodes, four hidden nodes and two output nodes.

V. THE CORRESPONDENCE BETWEEN FRNN AND TAKAGI-SUGENO TYPE FUZZY RULE BASE SYSTEM

Following [5] [4], it is easy to show that the FRNN is functionally equivalent to Takagi-Sugeno (TS) type fuzzy rule base system (FRBS) [18] under the following conditions:

1. The number of fuzzy rules for each class is equal to the number of the hidden nodes present in the FRNN.
2. The membership functions within each rule are chosen as Gaussian functions.
3. The t-norm operator used to compute each rule's firing strength is multiplication.

For 1-Class classification, the Table II shows how different parameters of the FRBS are related to the parameters of the FRNN. Thus, if we are able to train an FRNN with H hidden nodes and C output classes, then we can also construct a TS type FRBS with HC fuzzy if-then rules. Since the FRNN is a universal approximator, the derived FRBS is also a universal approximator.

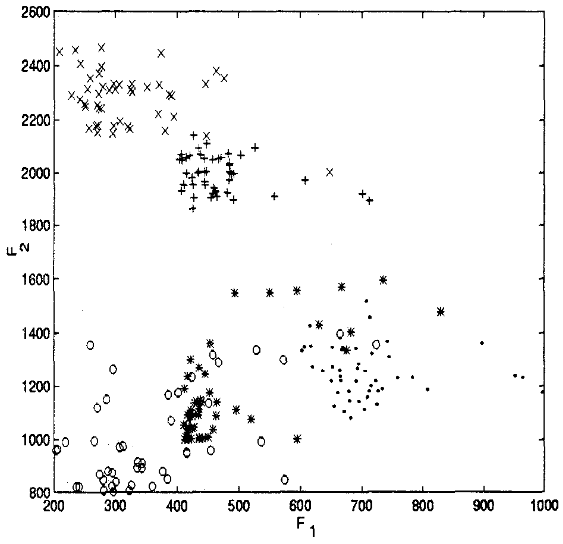

Fig. 2. Input speech data corresponding to the classes 'a', 'e', 'i', 'o' and 'u' are depicted in '.', '+', 'x', '*' and 'o', respectively. Along X-axis formant F_1 and along Y-axis formant F_2 are shown. The data contain both rough and fuzzy uncertainities.

TABLE II
THE MAPPING BETWEEN FUZZY RULE BASE SYSTEMS AND FUZZY-ROUGH NEURAL NETWORKS

FRBS		FRNN
No. of rules	=	No. of hidden nodes
Center of membership function for jth rule	=	Center of Gaussian function used in jth hidden node
Width of membership function for jth rule	=	Width of Gaussian function used in jth hidden node
Value of confidence factor for jth rule	=	Value of weight vector from jth hidden node to all output nodes

VI. RESULTS AND DISCUSSION

We study the task of vowel recognition [13] to demonstrate the efficiency of the proposed scheme. For our study, we consider the vowels 'a', 'e', 'i', 'o' and 'u'. The data required for the training are collected from the vowel part of the utterances of the consonant vowel pairs of three different speakers. The first three formants are used as features. The formants are extracted using linear prediction analysis [13]. The data set needed for this experiment is kept in the first author's homepage at
`http://www.geocities.com/SiliconValley/Lab/5073`
For ease of visualization, we illustrate (Fig. 2) the characteristic of the data on a two dimensional plane formed by the formants F_1 and F_2. From the figure, it is apparent that the classes 'e' and 'i' have lesser amount of fuzzy-rough ambiguity compared to the other three classes. We use the extracted features to constitute a training set of 230 examples. Here, our objective is to train various pattern classifiers as well as the FRNN on the same training set, and compare their classification performance on a different test set consisting of 1508 patterns.

First, we use Bayes classifier for multivariate normal patterns with the *a priori* probabilities $p_i = \frac{P_i}{P}$, where P_i denotes [10] the number of patterns in the ith class and P is the total number of training patterns. The covariance matrix for each class is determined from the training patterns of that particular class. Classification performance of the Bayes classifier on the test set is shown in the second column of the Table III. The Bayes classifier gives optimal classification performance for the probabilistic classifiers, provided the parameters of the input distribution are estimated from the inputs collected over the whole input space. In practice, the distribution parameters are estimated based only on a finite number of training data. As a result, the performance of the Bayes classifier is no longer optimal, but its performances approaches the optimal one as the number of input data is made very large (theoretically, it is infinity). Nevertheless, in the Table III, the Bayes classifier, based on a finite number of training samples, is used to compare the performance of the proposed method.

Next, we use the backpropagation (BP) learning algorithm to train feedforward neural networks with 3 input nodes, 5 output nodes and variable number of hidden nodes. The target patterns are 5 dimensional vectors containing 1 in one location and 0 in all others. Here, we adopt the strategy of picking the output node with the highest activation value as the output class corresponding to an input. The learning-rate is adaptively changed in the following way: If the error decreases during training, then the learning-rate is increased by a predefined amount. In contrast, if the error increases, then the learning-rate is decreased, and the new weights and errors are discarded. As a result, the error always decreases or stays as it is. The momentum is kept constant throughout the process. We used the BP learning algorithm on three different feedforward neural network architectures based on five, ten and fifteen hidden nodes. We have repeated the training of each architecture for 5000 iterations with 25 different weight and bias initializations. The mean classification performance of the BP algorithm over these three architectures are shown in the third, fourth and fifth columns of the Table III. Although for certain weight and bias initialization the overall classification performance is high, for some initializations the classification efficiency becomes quite low. Consequently, when the classification efficiency is averaged over twenty-five different initializations, the classification performance remains low.

Finally, we train the RBFNN and FRNN for the performance comparison. We use the supervised clustering scheme to cluster the input data. For that, we take all the training data from a particular class to determine the mean and variance of the clusters. Thus, we obtain one cluster for each class. The resultant FRNN and RBFNN have three input nodes, five hidden nodes and five output classes. For both the networks, the training data set is further used to train the weights between the hidden layer and the output layer. For the RBFNN, the weights in between the hidden and the output layer are determined directly by computation (i.e., by matrix inversion) so that we obtain the optimal weight and bias values. After training, both the networks are tested on the test set. The sixth and seventh columns of the Table III exhibit the performance of the RBFNN and FRNN. It can be observed that compared to all the other classifiers, the FRNN enhances the overall classification result significantly. For training and testing, the space and time complexity of the FRNN is also less.

Instead of using the FRNN for the classification, we can use the generated FRBS as well. From the cluster characteristics of the input data, we can construct the membership functions needed for each linguistic variable. The number of linguistic variables per input feature is 5 as there are 5 clusters. Since there are 5 clusters and 5 output classes, the number of possible rules is 25. The use of the rules makes it easy to understand the classification process. Moreover, we can ignore certain rules if the confidence factors associated with them are very low. If we fix a threshold as 0.001, then the rules with confidence factors less than 0.001 are deleted. The final class labels are decided based on the output node with maximum output. The rule pruning operation in the FRBS by fixing the threshold as 0.001, does not cause any damage to the classification performance (second column of the Table IV). When

TABLE III
Results of Vowel Classification for Different Types of Classification Algorithms are Shown in Percentage of Correct Classification

Class	Bayes classifier	Feedforward neural networks with BP algorithm			RBFNN	FRNN
		No. of hidden nodes				
		5	10	15		
'a'	84.89%	80.77%	69.49%	70.23%	86.10%	91.84%
'e'	82.86%	86.93%	83.28%	71.32%	67.83%	85.66%
'i'	87.33%	82.67%	80.30%	81.34%	88.50%	91.82%
'o'	40.29%	72.89%	76.56%	63.94%	63.59%	88.34%
'u'	89.86%	81.94%	79.92%	72.99%	86.93%	73.20%
overall	77.05%	81.04%	76.75%	71.97%	77.14%	86.17%

the threshold is chosen as 0.01, the FRBS is reduced to a set with eight rules. The classification performance of this FRBS does not degrade much as seen from the third column of the Table IV. If the threshold is chosen as 0.1, then the FRBS shrinks to a set of five rules, with one rule per class. Obviously, this is the minimum possible size of the rule base. Although the classification performance of the resultant FRBS degrades nearly by four percent (fourth column of the Table IV), the performance is still better than the Bayes, RBFNN and feedforward networks.

TABLE IV
Classification Performance of FRBS with Different Threshold Values

Class	Threshold value		
	0.001	0.01	0.1
'a'	91.84%	92.14%	92.44%
'e'	85.66%	85.66%	85.66%
'i'	91.82%	91.82%	91.82%
'o'	88.34%	87.86%	60.67%
'u'	73.20%	73.20%	84.31%
Overall	86.17%	86.13%	82.98%

If the input is [0,0,0], then the the input pattern is far away from the antecedent part of each rule. Hence, the activation of the antecedent parts of all the fuzzy rules are negligible. Consequently, the output given by each output node is almost zero indicating that the input pattern does not belong to any of the output classes. Therefore, the proposed network, as well as the derived FRBS, demonstrate the possibilistic classification capability.

VII. SUMMARY AND CONCLUSION

The main points of the paper can be summarised as follows: 1) This paper proposes the concept of fuzzy-rough neural networks for classification. 2) Proposed network attempts to capture the fuzzy linguistic ambiguity in the input data set as well as the rough uncertainty in the input-output relationship. 3) The working principle of the proposed network is based on the fuzzy-rough membership function. Since the fuzzy-rough membership function is possibilistic, the resultant network has possibilistic classification ability. 4) The proposed network can be directly mapped to a zeroth order TS type FRBS. Thus, unlike the feedforward networks with backpropagation algorithm, here the classification process is transparent. Moreover, the proposed network can be used to extract the fuzzy rules hidden inside the data. 5) Capturing fuzziness and roughness simultaneously indeed increases the classification performance in the vowel classification problem.

There are several extensions possible to the proposed FRNN. It is possible to use other t-norm operators in place of the multiplication operator used in the definition (7). However, this kind of generalization is possible if in the absence of fuzziness, the resultant function boils down to the rough membership function. Some of the possible t-norm operators are [6]: *drastic product*, *bounded product* and *logical product*. Moreover, it is possible to apply other aggregation operators like min and max in place of the summation operator. The exact choice of the operators will obviously be dictated by the application domain.

One interesting point to note in the FRBS derived from the FRNN is that the sum of the confidence factors used for each class (i.e., $\sum_j w_{jk}$) does not necessarily be equal to one. On the other hand, the sum of the confidence factors for each class used in the FRBS obtained from the RBFNN is essentially one [5]. Thus, the confidence factors used in the FRNN do not obey the addition rule used in the probability theory; rather, the confidence factors are like fuzzy measures [6], where the result of the addition depends on whether the confidence factors are *cooperative* or *noncooperative*. Therefore, one attractive issue here is to explore the possibility of using nonlinear operators, like fuzzy integrals [6], in place of the linear summation operator used in the definition (7).

If the output class is fuzzy, then it may be possible to make the FRNN more powerful by assigning the fuzzy memberships for the output class subjectively. For instance, if the output class for an image processing application is represented by BRIGHT, then it is possible to fit an 'S' fuzzy membership curve [10], and assign the fuzzy membership values for $\mu_{BRIGHT}(x)$ subjectively. However, if the domain specific knowledge is absent, then we have to be satisfied with the given crisp membership values for the output classes.

REFERENCES

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

[2] R. Dubes and A. Jain. *Algorithms that Cluster Data*. Prentice Hall, Englewood Cliffs, NJ, 1987.

[3] D. Dubois and H. Prade. Putting rough sets and fuzzy sets together. In R. Slowinski, editor, *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory*. Kluwer Academic Publishers, Dordrecht, 1992.

[4] K. J. Hunt, R. Haas, and R. Murray-Smith. Extending the functional equivalence of radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 7(3):776–771, May 1996.

[5] J. S. R. Jang, C. T. Sun, and E. Mijutani. *Neuro-Fuzzy and Soft Computing*. Prentice-Hall, Englewood Cliffs, NJ, 1997.

[6] J. M. Keller, P. Gader, H. Tahani, J. H. Chiang, and M. Mohamed. Advances in fuzzy integration for pattern recognition. *Fuzzy Sets and Systems*, (65):273–283, 1994.

[7] G. S. Klir and B. Yuan. *Fuzzy sets and Fuzzy Logic – Theory and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[8] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, May 1993.

[9] N. R. Pal and J. C. Bezdek. On cluster validity for the fuzzy C-means model. *IEEE Transactions on Fuzzy Systems*, 3(3):330–379, August 1995.

[10] S. K. Pal and D. Dutta Majumder. *Fuzzy Mathematical Approach to Pattern Recognition*. Wiley (Halsted Press), New York, 1986.

[11] Z. Pawlak. Rough sets. *International Journal of Computer and Information Science*, 11:341–356, 1982.

[12] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer, Dordrecht, 1991.

[13] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliff, NJ, 1993.

[14] M. Sarkar. Evolutionary programming based fuzzy clustering. In *Proceedings of Fifth Annual Conference on Evolutionary Programming (San Diego)*, pages 247–256, 1996.

[15] M. Sarkar and B. Yegnanarayana. Rough-fuzzy membership functions in classification. Accepted in *Fuzzy Sets and Systems*.

[16] M. Sarkar and B. Yegnanarayana. Rough-fuzzy membership functions. In *Proceedings of IEEE International Conference on Fuzzy Systems (Anchorage, Alaska, USA)*, May 4-9 1998.

[17] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[18] H. Takagi and M. Sugeno. Fuzzy identification of systems and applications to modeling and control. *IEEE Transactions on System, Man and Cybernetics*, 15:116–132, January/February 1985.

[19] S. K. M. Wong and W. Ziarko. Comparison of the probabilistic approximate classification and fuzzy set model. *Fuzzy Sets and Systems*, 21:357–362, 1987.

[20] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.