Rough Fuzzy MLP: Knowledge Encoding and Classification

Mohua Banerjee, Sushmita Mitra, and Sankar K. Pal, *Fellow, IEEE*

***Abstract*— A new scheme of knowledge encoding in a fuzzy multilayer perceptron (MLP) using rough set-theoretic concepts is described. Crude domain knowledge is extracted from the data set in the form of rules. The syntax of these rules automatically determines the appropriate number of hidden nodes while the dependency factors are used in the initial weight encoding. The network is then refined during training. Results on classification of speech and synthetic data demonstrate the superiority of the system over the fuzzy and conventional versions of the MLP (involving no initial knowledge).**

***Index Terms*— Fuzzy MLP, knowledge-based networks, network design, pattern recognition, rough sets, rule generation, soft computing, speech recognition.**

I. Introduction

THERE has recently been a spurt of activity to integrate different computing paradigms such as fuzzy set theory, neural networks, genetic algorithms, and rough set theory, for generating more efficient hybrid systems that can be classified as *soft computing* methodologies [1], [2]. The purpose is to provide flexible information processing systems that can exploit the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth in order to achieve tractability, robustness, and low cost in real-life ambiguous situations [3].

Neuro-fuzzy computing [4], [5] capturing the merits of fuzzy set theory [6] and artificial neural networks (ANN's) [7], constitutes one of the best-known hybridizations encompassed in soft computing. This integration promises to provide, to a great extent, more intelligent systems (in terms of parallelism, fault tolerance, adaptivity, and uncertainty management) to handle real-life recognition/decision making problems. The fuzzy multilayer perceptron (MLP) [8], [9] is such an example which incorporates fuzzy set-theoretic concepts at the input and output levels and during learning. It is found to be more efficient than the conventional MLP for classification and rule generation.

Generally, ANN's consider a fixed topology of neurons connected by links in a predefined manner. These connection weights are usually initialized by small random values. Knowledge-based networks [10], [11] constitute a special class of ANN's that consider crude domain knowledge to generate the initial network architecture which is later refined in the presence of training data. This process helps in reducing the searching space and time while the network traces the optimal solution. Node growing and link pruning are also made in order to generate the optimal network architecture. In this paper, we demonstrate how the theory of rough sets can be utilized for extracting domain knowledge.

The theory of rough sets [12] has recently emerged as another major mathematical approach for managing uncertainty that arises from inexact, noisy, or incomplete information. It has been investigated in the context of expert systems, decision support systems, machine learning, inductive learning and various other areas of application. It is found to be particularly effective in the area of knowledge reduction. The focus of rough set theory is on the ambiguity caused by limited discernibility of objects in the domain of discourse. The intention is to approximate a *rough* (imprecise) concept in the domain of discourse by a pair of *exact* concepts, called the lower and upper approximations. These exact concepts are determined by an *indiscernibility* relation on the domain, which, in turn, may be induced by a given set of *attributes* ascribed to the objects of the domain. These approximations are used to define the notions of *discernibility matrices*, *discernibility functions* [13], *reducts*, and *dependency factors* [12], all of which play a fundamental role in the reduction of knowledge.

Many have looked into the implementation of decision rules extracted from operation data using rough set formalism, especially in problems of machine learning from examples and control theory [14]. In the context of neural networks, an attempt of such implementation has been made by Yasdi [15]. The intention was to use rough sets as a tool for structuring the neural networks. The methodology consisted of generating rules from training examples by rough-set learning, and mapping the dependency factors of the rules into the connection weights of a four-layered neural network. Application of rough sets in neurocomputing has also been made in [16]. However, in this method, rough sets were used for knowledge discovery at the level of data acquisition, (*viz.*, in preprocessing of the feature vectors), and not for structuring the network.

In this article, we have attempted to integrate rough sets and fuzzy neural network for designing a knowledge-based system.

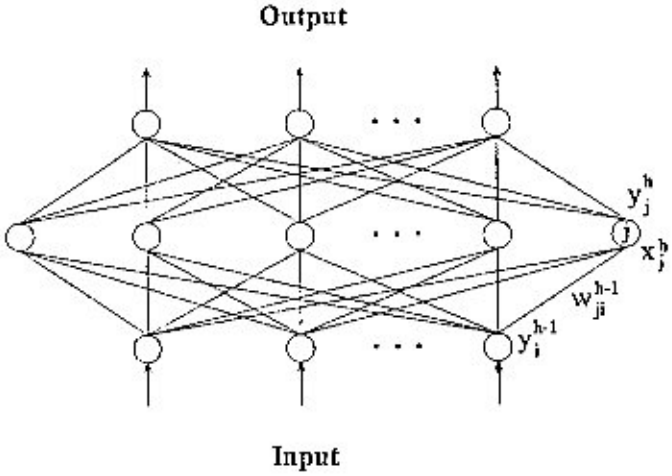


Fig. 1. Three-layered MLP.

Rough set-theoretic techniques are utilized for extracting crude domain knowledge, that is then encoded among the connection weights. Methods are derived to model 1) convex decision regions with single object representatives and 2) arbitrary decision regions with multiple object representatives. A three-layered fuzzy MLP [8] is considered. The input is modeled in terms of the $3n$-dimensional linguistic feature space while the output consists of class membership values. The feature space gives us the condition attributes and the output classes the decision attributes, so as to result in a decision table. This table, however, may be transformed, keeping the complexity of the network to be constructed in mind. Rules are then generated from the (transformed) table by computing relative reducts. The dependency factors of these rules are encoded as the initial connection weights of the fuzzy MLP. The network is next trained to refine its weight values.

The knowledge encoding procedure, unlike most other methods [10], [11], involves a nonbinary weighting mechanism based on a detailed and systematic estimation of the available domain information. It may be noted that the appropriate number of hidden nodes is automatically determined. The classification performance is found to be better than the conventional and fuzzy versions of the MLP. The model is capable of handling input in numerical, linguistic and set forms, and can tackle uncertainty due to overlapping classes.

A brief description of the fuzzy MLP used is provided in Section II. The basics of rough set theory are presented in Section III. In Section IV, we describe the knowledge encoding methodology. The model is implemented on real-life speech data as well as synthetic data (in Section V) for classification. Comparison is provided with the standard Bayes' classifier, k-nearest neighbors (k-NN) classifier, classification and regression tree [17], and the conventional and fuzzy versions of the MLP (involving no initial knowledge). The paper is concluded in Section VI.

II. Fuzzy MLP Model

In this section we describe, in brief, the fuzzy MLP [8] which is used for designing the knowledge-based network. Consider the layered network given in Fig. 1. The output of a neuron in any layer (h) other than the input layer $(h = 0)$ is given as

$$y_j^{(h)} = \frac{1}{1 + \exp\left(-\sum_i y_i^{(h-1)} w_{ji}^{(h-1)}\right)} \tag{1}$$

where $y_i^{(h-1)}$ is the state of the ith neuron in the preceding $(h-1)$th layer and $w_{ji}^{(h-1)}$ is the weight of the connection from the ith neuron in layer $(h-1)$ to the jth neuron in layer (h). For nodes in the input layer, $y_j^{(0)}$ corresponds to the jth component of the input vector. Note that $x_j^{(h)} = \sum_i y_i^{(h-1)} w_{ji}^{(h-1)}$, as depicted in Fig. 1. The mean square error in output vectors is minimized by the backpropagation algorithm using a gradient descent with a gradual decrease of the gain factor.

A. Input Vector

An n-dimensional pattern $\mathbf{F}_i = [F_{i1}, F_{i2}, \cdots, F_{in}]$ is represented as a $3n$-dimensional vector [18]

$$\begin{aligned}\mathbf{F}_i &= [\mu_{low(F_{i1})}(\mathbf{F}_i), \cdots, \mu_{high(F_{in})}(\mathbf{F}_i)] \\ &= [y_1^{(0)}, y_2^{(0)}, \cdots, y_{3n}^{(0)}]\end{aligned} \tag{2}$$

where the μ values indicate the membership functions of the corresponding linguistic π-sets *low*, *medium*, and *high* along each feature axis and $y_1^{(0)}, \cdots, y_{3n}^{(0)}$ refer to the activations of the $3n$ neurons in the input layer. The three overlapping π-sets along a feature axis are depicted in Fig. 2.

When the input feature is numerical, we use the π-fuzzy sets (in the one-dimensional form), with range [0, 1], represented as

$$\pi(F_j; c, \lambda) = \begin{cases} 2\left(1 - \dfrac{\|F_j - c\|}{\lambda}\right)^2, & \text{for } \dfrac{\lambda}{2} \le \|F_j - c\| \le \lambda \\ 1 - 2\left(\dfrac{\|F_j - c\|}{\lambda}\right)^2, & \text{for } 0 \le \|F_j - c\| \le \dfrac{\lambda}{2} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\lambda(>0)$ is the radius of the π-function with c as the central point.

When the input feature F_j is linguistic, its membership values for the π-sets *low* (L), *medium* (M), and *high* (H) are quantified as

$$\text{low} \equiv \left\{ \frac{0.95}{L}, \frac{\pi\left(F_j\left(\frac{0.95}{L}\right); c_{j_m}, \lambda_{j_m}\right)}{M} \right. \left. \frac{\pi\left(F_j\left(\frac{0.95}{L}\right); c_{j_h}, \lambda_{j_h}\right)}{H} \right\}$$

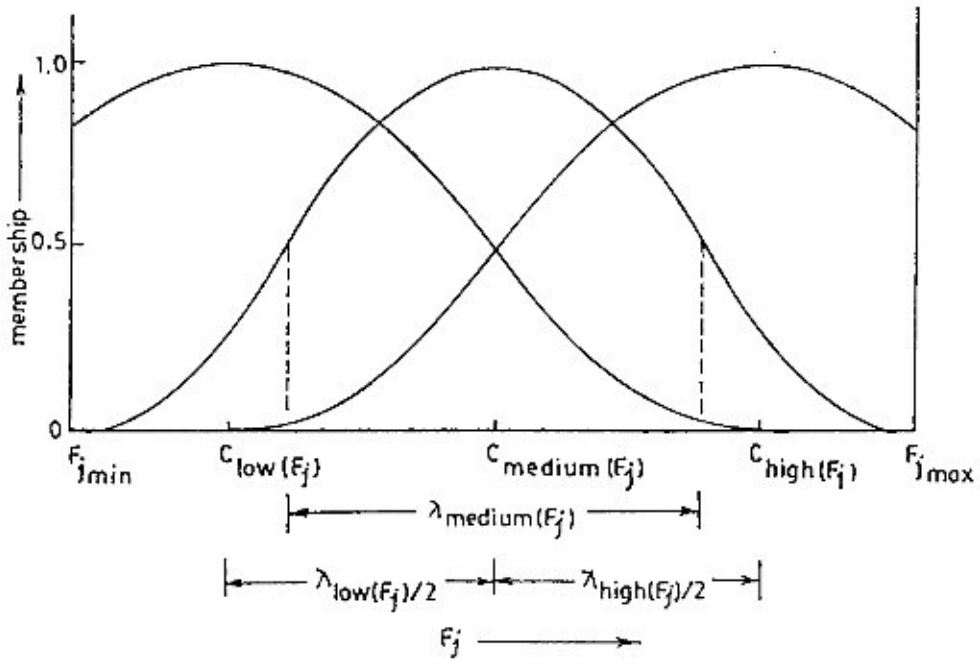


Fig. 2. Overlapping linguistic Π-sets.

$$\text{medium} \equiv \left\{ \frac{\pi\left(F_j\left(\frac{0.95}{M}\right); c_{j_m}, \lambda_{j_m}\right)}{L} \cdot \frac{0.95}{M} \right. \left. \frac{\pi\left(F_j\left(\frac{0.95}{M}\right); c_{j_m}, \lambda_{j_m}\right)}{H} \right\}$$

$$\text{high} = \left\{ \frac{\pi\left(F_j\left(\frac{0.95}{H}\right); c_{j_h}, \lambda_{j_h}\right)}{L} \right. \left. \frac{\pi\left(F_j\left(\frac{0.95}{H}\right); c_{j_h}, \lambda_{j_h}\right)}{M} \cdot \frac{0.95}{H} \right\}$$

where c_{j_l}, λ_{j_l}, c_{j_m}, λ_{j_m}, c_{j_h}, λ_{j_h} indicate the centers and radii of the three linguistic properties along the jth axis, and $F_j(0.95/L)$, $F_j(0.95/M)$, $F_j(0.95/H)$ denote the corresponding feature values F_j at which the three linguistic properties attain membership values of 0.95.

For example, the linguistic feature *low* is represented by three components corresponding to the membership values of the three π-sets *low* (L), *medium* (M) and *high* (H) (Fig. 2). $0.95/L$ means a membership of 0.95 for L. $\pi(F_j(0.95/L); c_{j_m}, \lambda_{j_m})/M$ refers to the membership attained by the π-set M for that F_j which caused π-set L to have a membership value of 0.95. Similarly, $\pi(F_j(0.95/L); c_{j_h}, \lambda_{j_h})/H$ refers to the membership attained by the π-set H for that F_j which caused π-set L to have a membership value of 0.95.

B. Output Representation

Consider an l-class problem domain such that we have l nodes in the output layer. Let the n-dimensional vectors $\mathbf{o}_k = [o_{k1} \cdots o_{kn}]$ and $\mathbf{v}_k = [v_{k1} \cdots v_{kn}]$ denote the mean and standard deviation, respectively, of the numerical training data for the kth class c_k. The weighted distance of the training pattern $\mathbf{F}_i$ from the kth class c_k is defined as

$$z_{ik} = \sqrt{\sum_{j=1}^{n} \left[\frac{F_{ij} - o_{kj}}{v_{kj}}\right]^2}, \qquad \text{for } k = 1, \cdots, l \tag{4}$$

where F_{ij} is the value of the jth component of the ith pattern point.

The membership of the ith pattern in class k, lying in the range [0, 1], is defined as [19]

$$\mu_k(\mathbf{F}_i) = \frac{1}{1 + \left(\frac{z_{ik}}{f_d}\right)^{f_e}} \tag{5}$$

where positive constants f_d and f_e are the denominational and exponential fuzzy generators controlling the amount of fuzziness in this class-membership set.

Then, for the ith input pattern, the desired output of the jth output node is defined as

$$d_j = \mu_j(\mathbf{F}_i). \tag{6}$$

According to this definition a pattern can simultaneously belong to more than one class, and this is determined from the training set used during the learning phase.

III. Rough Set Preliminaries

Let us present some requisite preliminaries of rough set theory. For details one may refer to [12] and [13].

An *information system* is a pair $\mathcal{S} = \langle U, A \rangle$, where U is a nonempty finite set called the *universe* and A a nonempty finite set of *attributes*. An attribute a can be regarded as a function from the domain U to some value set V_a.

An information system may be represented as an *attribute-value table*, in which rows are labeled by objects of the universe and columns by the attributes.

With every subset of attributes $B \subseteq A$, one can easily associate an equivalence relation I_B on U:

$$I_B = \{(x, y) \in U: \text{for every } a \in B,\ a(x) = a(y)\}.$$

Then $I_B = \bigcap_{a \in B} I_a$.

If $X \subseteq U$, the sets $\{x \in U: [x]_B \subseteq X\}$ and $\{x \in U: [x]_B \cap X \neq \emptyset\}$, where $[x]_B$ denotes the equivalence class of the object $x \in U$ relative to I_B, are called the *B-lower* and *B-upper approximation* of X in $\mathcal{S}$ and denoted $\underline{B}X$, $\overline{B}X$, respectively.

$X (\subseteq U)$ is *B-exact* or *B-definable* in $\mathcal{S}$ if $\underline{B}X = \overline{B}X$. It may be observed that $\underline{B}X$ is the greatest B-definable set contained in X, and $\overline{B}X$ is the smallest B-definable set containing X. Let us consider the following simple example.

Consider an *information system* $\langle U, \{a\} \rangle$, where the domain U consists of the students of a school, and there is a single attribute a—that of "belonging to a class." Then U is partitioned by the classes of the school.

Now take the situation when an infectious disease has spread in the school, and the authorities take the two following steps.

1) If at least one student of a class is infected, all the students of that class are vaccinated. Let $\overline{B}$ denote the union of such classes.
2) If every student of a class is infected, the class is temporarily suspended. Let $\underline{B}$ denote the union of such classes.

Then $\underline{B} \subseteq \overline{B}$. Given this information, let the following problem be posed: *Identify the collection of infected students.*

Clearly, there cannot be a unique answer. But any set I that is given as an answer, must contain $\underline{B}$ *and* at least one student from each class comprising $\overline{B}$.

In other words, it must have $\underline{B}$ as its *lower approximation* and $\overline{B}$ as its *upper approximation.*

I is then a *rough* concept/set in the information system $\langle U, \{a\} \rangle$.

Further, it may be observed that any set I' given as another answer, is *roughly equal* to I, in the sense that both are represented (characterized) by $\overline{B}$ and $\underline{B}$.

We now define the notions relevant to knowledge reduction. The aim is to obtain irreducible but essential parts of the knowledge encoded by the given information system—these would constitute *reducts* of the system. So one is, in effect, looking for *maximal* sets of attributes taken from the initial set (A, say), which induce the *same* partition on the domain as A. In other words, the essence of the information remains intact, and superfluous attributes are removed. Reducts have been nicely characterized in [13] by *discernibility matrices* and *discernibility functions.* A principal task in our proposed methods will be to compute reducts relative to a particular kind of information system, and relativized versions of these matrices and functions shall be the basic tools used in the computation.

Let $U = \{x_1 \cdots x_n\}$ and $A = \{a_1 \cdots a_m\}$ in the information system $\mathcal{S} = \langle U, A \rangle$. By the discernibility matrix [denoted $\mathbf{M}(\mathcal{S})$] of $\mathcal{S}$ is meant an $n \times n$-matrix such that

$$c_{ij} = \{a \in A: a(x_i) \neq a(x_j)\}, \qquad i, j = 1, \cdots, n. \quad (7)$$

A discernibility function $f_{\mathcal{S}}$ is a Boolean function of m Boolean variables $\overline{a}_1, \cdots, \overline{a}_m$ corresponding to the attributes $a_1, \cdots, a_m$, respectively, and defined as follows:

$$f_{\mathcal{S}}(\overline{a}_1, \cdots, \overline{a}_m) = \bigwedge \left\{ \bigvee (c_{ij}): 1 \leq j < i \leq n,\ c_{ij} \neq \emptyset \right\} \quad (8)$$

where $\bigvee(c_{ij})$ is the disjunction of all variables $\overline{a}$ with $a \in c_{ij}$. It is seen in [13] that $\{a_{i_1}, \cdots, a_{i_p}\}$ is a reduct in $\mathcal{S}$ if and only if $a_{i_1} \wedge \cdots \wedge a_{i_p}$ is a prime implicant (constituent of the disjunctive normal form) of $f_{\mathcal{S}}$.

The next concept that we shall require during rule generation, is that of *dependency factor*. It may well happen for $B, C \subseteq A$, that C *depends* on B, i.e., $I_B \subseteq I_C$—so that information due to the attributes in C is derivable from that due to the attributes in B. This dependency can be partial, in which case one introduces a dependency factor df, $0 \leq df \leq 1$

$$df = \frac{\text{card}(\text{POS}_B(C))}{\text{card}(U)} \quad (9)$$

where $\text{POS}_B(C) = \bigcup_{X \in I_C} \underline{B}X$, and card denotes cardinality of the set.

We are concerned with a specific type of information system $\mathcal{S} = \langle U, A \rangle$, called a *decision table.* The attributes in such a system are distinguished into two parts, *viz. condition* and *decision* attributes. Classification of the domain due to decision attributes could be thought of as that given by an expert. One may now want to deal with *consistent* decision tables, such that a decision attribute does not assign more than one value to an object, or for that matter, to objects indiscernible from each other with respect to the given (condition) attributes. Formally we have the following.

Let $C, D \subseteq A$ be the sets of condition and decision attributes of $\mathcal{S}$, respectively. The *rank* of a decision attribute $d \in D$, $r(d)$, is the cardinality of the image $d(U)$ of the function d on the value set V_d. One can then assume that $V_d = \{1, \cdots, r(d)\}$.

The *generalized decision* in $\mathcal{S}$ corresponding to d is then defined as a function $\partial_{\mathcal{S}}: U \to \mathcal{P}(\{1, \cdots, r(d)\})$ such that $\partial_{\mathcal{S}}(x) = \{i: \exists x' \in [x]_C \text{ and } d(x') = i\}$, $\mathcal{P}$ denoting the power set. A decision table $\mathcal{S}$ with $D = \{d\}$ is called *consistent (deterministic)* if $\text{card}(\partial_{\mathcal{S}}(x)) = 1$ for any $x \in U$, or equivalently, if and only if $\text{POS}_C(d) = U$. Otherwise, $\mathcal{S}$ is *inconsistent (nondeterministic).*

Knowledge reduction now consists of eliminating superfluous values of the condition attributes by computing their reducts, and we come to the notion of a *relative reduct.*

An attribute $b \in B (\subseteq C)$ is D-dispensable in B, if $\text{POS}_B(D) = \text{POS}_{B \setminus \{b\}}(D)$; otherwise b is D-indispensable in B. If every attribute from B is D-indispensable in B, B is D-independent in $\mathcal{S}$. A subset B of C is a D-reduct in $\mathcal{S}$ if B is D-independent in $\mathcal{S}$ and $\text{POS}_C(D) = \text{POS}_B(D)$.

Relative reducts can be computed by using a *D-discernibility matrix*. If $U = \{x_1, \cdots, x_n\}$, it is an $n \times n$ matrix [denoted $\mathbf{M}_D(\mathcal{S})$], the ijth component of which has the form

$$c_{ij} = \{a \in C: a(x_i) \neq a(x_j) \text{ and } (x_i, x_j) \notin I_D\} \quad (10)$$

for $i, j = 1, \cdots, n$.

The relative discernibility function f_D is constructed from the D-discernibility matrix in an analogous way as f_S from the discernibility matrix of $\mathcal{S}$ [cf. (7) and (8)]. It is once more observed that [13] $\{a_{i_1} \cdots, a_{i_p}\}$ is a D-reduct in $\mathcal{S}$ if and only if $a_{i_1} \wedge \cdots \wedge a_{i_p}$ is a prime implicant of f_D.

IV. Network Configuration Using Rough Sets

Here we formulate two methods for rule generation and knowledge encoding for configuring a network. Method I works on the assumption that each object of the domain of discourse corresponds to a single decision attribute. On the other hand, Method II is able to deal with multiple objects corresponding to one decision attribute. From the perspective of pattern recognition, this implies using a single prototype to model a (convex) decision region in case of Method I. For Method II, this means using multiple prototypes to serve as representatives of any arbitrary decision region.

The crude domain knowledge, so extracted, is encoded among the connection weights, leading to the design of a knowledge-based network. Such a network is found to be more efficient than the conventional versions for the following reason. During learning an MLP searches for the set of connection weights that corresponds to some local minima. In other words, it searches for that set of weights that minimizes the difference between the target vector and the actual output (obtained by the MLP). Note that there may be a large number of such minimum values corresponding to various *good* solutions. If we initially set these weights so as to be near one such *good* solution, the searching space may be reduced and learning thereby becomes faster. The architecture of the network becomes simpler due to the inherent reduction of the redundancy among the connection weights.

A block diagram in Fig. 3 illustrates the entire procedure for both the methods.

A. Method I

Let $\mathcal{S} = < U, A >$ be a decision table, with C and D its sets of condition and decision attributes, respectively. In this method we assume that there is a decision attribute $d_i \in D$ corresponding to each object $x_i \in U$, in the sense that all objects other than x_i are indiscernible with respect to d_i.

1) Rule Generation: For each D-reduct $B = \{b_1, \cdots, b_k\}$ (say), we define a discernibility matrix [denoted $\mathbf{M}_D(B)$] from the D-discernibility matrix [given by (10)] as follows:

$$c_{ij} = \{a \in B: a(x_i) \neq a(x_j)\} \quad (11)$$

for $i, j = 1, \cdots, n$.

Now for each object x_i of U, we consider the discernibility function $f_D^{x_i}$ which is defined as

$$f_D^{x_i} = \bigwedge \left\{ \bigvee (c_{ij}): 1 \leq j \leq n, j \neq i, c_{ij} \neq \emptyset \right\} \quad (12)$$

where $\bigvee(c_{ij})$ is the disjunction of all members of c_{ij}.

$f_D^{x_i}$ is brought to its conjunctive normal form (CNF) P_i. For $i = 1, \cdots, n$, $f_D^{x_i}$ then gives rise to a dependency rule r_i, *viz.* $P_i \rightarrow d_i$, where $d_i \in D$ corresponds to the object x_i.

It may be noticed that each component of P_i induces an equivalence relation on U as follows. If a component is a single attribute b, the relation I_b is taken. If a component of the CNF is a disjunct of attributes, say $b_{i_1}, \cdots, b_{i_n} \in B$, we consider the transitive closure of the union of the relations $I_{b_{i_1}}, \cdots, I_{b_{i_n}}$. Let I_i denote the intersection of all these equivalence relations.

The dependency factor df_i for r_i is then given by

$$df_i = \frac{\text{card}(\text{POS}_i(d_i))}{\text{card}(U)} \quad (13)$$

where $\text{POS}_i(d_i) = \bigcup_{X \subset I_{d_i}} l_i(X)$, and $l_i(X)$ is the lower approximation of X with respect to I_i.

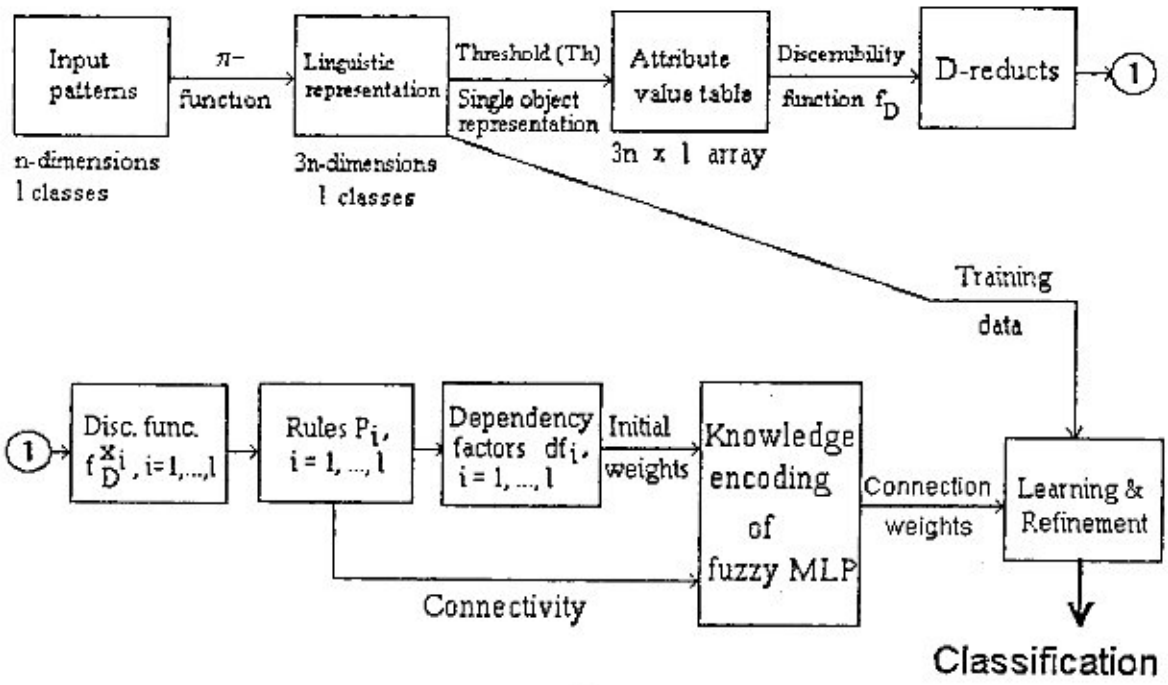
2) Knowledge Encoding: Here, we formulate a methodology for encoding initial knowledge in the fuzzy MLP of [8], following the above algorithm.

Let us consider the case of feature F_j for class c_k in the l-class problem domain. The inputs for the ith representative sample $\mathbf{F}_i$ are mapped to the corresponding three-dimensional feature space of $\mu_{\text{low}(F_{ij})}(\mathbf{F}_i)$, $\mu_{\text{medium}(F_{ij})}(\mathbf{F}_i)$, and $\mu_{\text{high}(F_{ij})}(\mathbf{F}_i)$, by (2). Let these be represented by L_j, M_j, and H_j, respectively. We consider only those attributes which have a numerical value greater than some threshold Th ($0.5 \leq Th < 1$). This implies clamping those features demonstrating high membership values with a one, while the others are fixed at zero. In this manner an $l \times 3n$-dimensional attribute-value (decision) table can be generated from the n-dimensional data set.

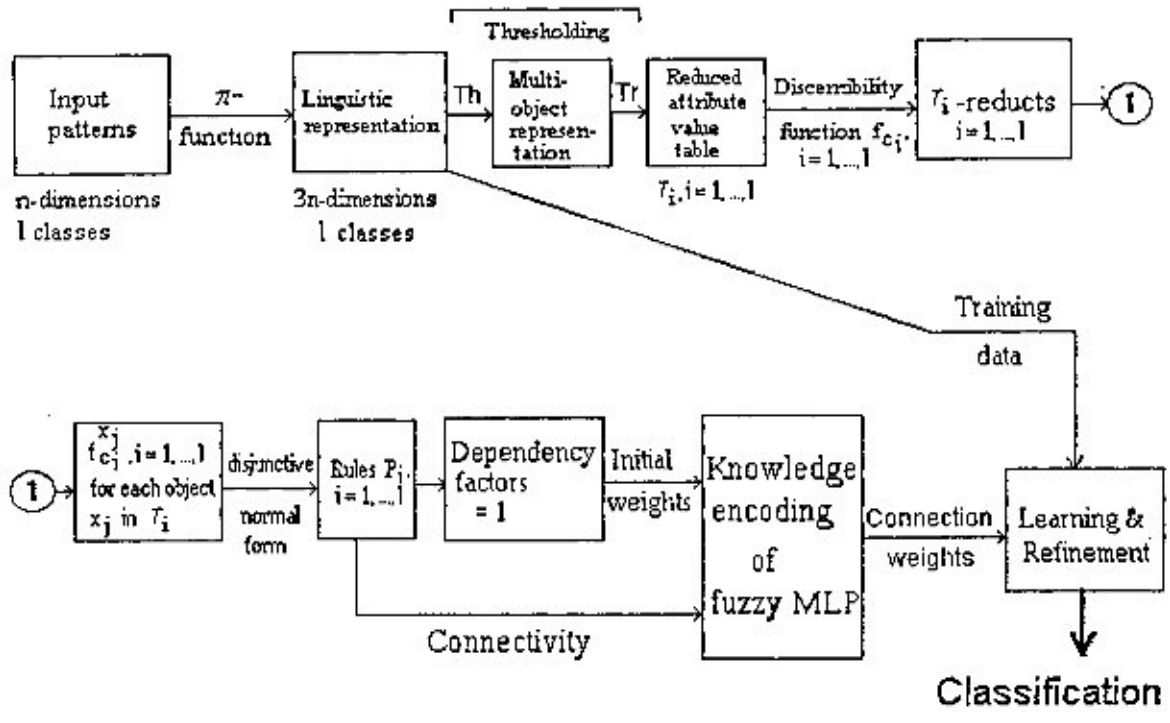
As sketched in the previous section, one generates the dependency rules for each of the l classes, such that the antecedent part contains a subset of the $3n$ attributes, along with the corresponding dependency factors.

Let us now design the initial structure of the three-layered fuzzy MLP. The input layer consists of the $3n$ attribute values and the output layer is represented by the l classes. The hidden layer nodes model the disjuncts ($\vee$) in the antecedents of the dependency rules. For each disjunct, corresponding to one output class (one dependency rule), we dedicate one hidden node. Only those input attributes that appear in a disjunct are connected to the appropriate hidden node, which in turn is connected to the corresponding output node. Each conjunct ($\wedge$) is modeled at the output layer by joining the corresponding hidden nodes. Note that a single attribute (involving no disjuncts) is directly connected to the appropriate output node via a hidden node.

Next we proceed to the description of the initial weight encoding procedure. Let the dependency factor for a particular dependency rule for class c_k be α by (13). The weight $w_{ki}^{(1)}$ between a hidden node i and output node k is set at



(a)



(b)

Fig. 3. Block diagram of the rule generation and knowledge encoding procedure. (a) Method I. (b) Method II.

$\alpha/(fac) + \epsilon$, where fac refers to the number of conjunctions in the antecedent of the rule and ϵ is a small random number taken to destroy any symmetry among the weights. Note that $fac \geq 1$ and each hidden node is connected to only one output node. Let the initial weight so clamped at a hidden node be denoted as β. The weight $w_{ia_j}^{(0)}$ between an attribute a_j [where a corresponds to low (L), medium (M) or high (H)] and hidden node i is set to $\beta/(facd) + \epsilon$, such that $facd$ is the number of attributes connected by the corresponding disjunct. Note that $facd \geq 1$. The sign of the weight is set to positive (negative) if the corresponding entry in row k, column a_j is 1 (0). Thus, for an l-class problem domain we have at least l hidden nodes. All other possible connections in the resulting fuzzy MLP are set as small random numbers. It is to be mentioned that the number of hidden nodes is determined from the dependency rules.

The connection weights, so encoded, are then refined by training the network on the pattern set supplied as input.

B. Method II

Let $\mathcal{S} =< U, A >$ be a decision table, with C and $D = \{d_1, \cdots, d_n\}$ its sets of condition and decision attributes, respectively.

1) Rule Generation: We divide the decision table $\mathcal{S} =< U, A >$ into n tables $\mathcal{S}_i =< U_i, A_i >$, $i = 1, \cdots, n$, corresponding to the n decision attributes $d_1, \cdots, d_n$, where

$$U = U_1 \cup \cdots \cup U_n \quad \text{and} \quad A_i = C \cup \{d_i\}.$$

The size of each $\mathcal{S}_i$ $(i = 1, \cdots, n)$ is first reduced with the help of a threshold on the number of occurrences of the same pattern of attribute values. This will be elicited in the sequel. Let the reduced decision table be denoted by $\mathcal{T}_i$, and $\{x_{i_1}, \cdots, x_{i_p}\}$ be the set of those objects of U_i that occur in $\mathcal{T}_i, i = 1 \cdots, n$.

Using (11) and (12), for each d_i-reduct B (say), we define the discernibility matrix $(\mathbf{M}_{d_i}(B))$ and for every object $x_j \in \{x_{i_1}, \cdots, x_{i_p}\}$, the discernibility function $f_{d_i}^{x_j}$. Then $f_{d_i}^{x_j}$ is brought to its CNF One thus obtains a dependency rule r_i, *viz.* $P_i \rightarrow d_i$, where P_i is the disjunctive normal form (DNF) of $f_{d_i}^{x_j}$, $j \in \{i_1, \cdots, i_p\}$. It may then be noticed that the dependency factor df_i for each r_i is one [by (13)].

2) Knowledge Encoding: The knowledge encoding scheme is similar to that described in Section IV-A. As this method considers multiple objects in a class (unlike Method I), we generate a separate $n_k \times 3n$-dimensional attribute value table for each class c_k (where n_k indicates the number of objects in c_k).

Let there be m sets $O_1, \cdots, O_m$ of objects in the table having identical attribute-values, and $\text{card}(O_i) = n_{k_i}$, $i = 1, \cdots, m$, such that $n_{k_1} \geq \cdots \geq n_{k_m}$ and $\sum_{i=1}^{m} n_{k_i} = n_k$. The attribute-value table can now be represented as an $m \times 3n$ array. Let $n_{k'_1}, n_{k'_2}, \cdots, n_{k'_m}$ denote the distinct elements among $n_{k_1}, \cdots, n_{k_m}$ such that $n_{k'_1} > n_{k'_2} > \cdots > n_{k'_m}$. We apply a heuristic threshold function defined by

$$Tr = \left\lceil \frac{\sum_{i=1}^{m} \frac{1}{n_{k'_i} - n_{k'_{i+1}}}}{Th} \right\rceil. \tag{14}$$

All entries having frequency less than Tr are eliminated from the table, resulting in the reduced attribute-value table. Note that the main motive of introducing this threshold function lies in reducing the size of the resulting network. We attempt to eliminate noisy pattern representatives (having lower values of n_{k_i}) from the reduced attribute-value table. The whole approach is, therefore, data dependent. The dependency rule for each class is obtained by considering the corresponding reduced attribute-value table. A smaller table leads to a simpler rule in terms of conjunctions and disjunctions, which is then translated into a network having fewer hidden nodes. The objective is to strike a balance by reducing the network complexity and reaching a *good* solution, perhaps at the expense of not achieving the *best* performance.

While designing the initial structure of the fuzzy MLP, we consider the union of the rules of the l classes. Here the hidden layer nodes model the first level (innermost) operator in the antecedent part of a rule, which can be either a conjunct or a disjunct. The output layer nodes model the outer level operator, which can again be either a conjunct or a disjunct. As mentioned earlier, the dependency factor of any rule is one in this method. The initial weight encoding procedure is the same as described before. Since each class has multiple objects, the sign of the weight is set randomly.

V. Implementation and Experimental Results

Here we implement the two methods on real-life and artificial data. The initial weight encoding scheme is demonstrated and recognition scores are presented. The data sets are available on the internet at http://isical.ac.in/sushmita/data/vowsy.html.

The speech data **Vowel** [20] deals with 871 Indian Telugu vowel sounds. These were uttered in a consonant-vowel-consonant context by three male speakers in the age group of 30–35 years. The data set (depicted in two dimensions for ease of understanding) has three features: F_1, F_2, and F_3 corresponding to the first, second, and third vowel formant frequencies obtained through spectrum analysis of the speech data. Fig. 4 provides the projection in the F_1–F_2 plane, depicting the six vowel classes—∂, a, i, u, e, o. These overlapping classes shall be denoted by $c_1, \cdots, c_6$, respectively, in the sequel.

The synthetic data **Pat** consists of 880 pattern points in the two-dimensional space F_1–F_2, as depicted in Fig. 5. There are three linearly nonseparable pattern classes. The figure is marked with classes 1 (c_1) and 2 (c_2), while class 3 (c_3) corresponds to the background region.

The training set considered 50% of the data selected randomly from each of the pattern classes. The remaining 50% data constituted the test set. It is found that the knowledge-based model converges to a good solution with a small number of training epochs (iterations) in both cases. Note that the Vowel data consists of convex classes which may be modeled by single representative points (objects). However, the synthetic data set Pat consists of concave and disjoint classes that can only be modeled by multiple representative points (objects). As Method I considers single object classes only, the synthetic data could not be used there. On the other hand, both data sets are used in Method II which considers multiple objects in a class.

A. Method I

The rough set-theoretic technique is applied on the vowel data to extract some knowledge which is initially encoded among the connection weights of the fuzzy MLP. The data is first transformed into the 3 -dimensional linguistic space of (2). A threshold of $Th = 0.8$ is imposed on the resultant input components such that $y_i^{(0)} = 1$ if $y_i^{(0)} \geq 0.8$ and zero

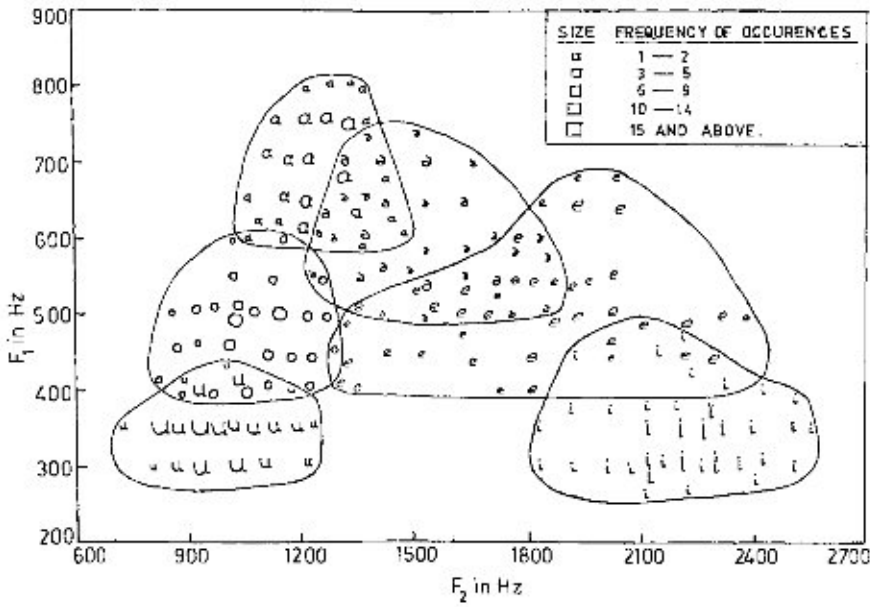


Fig. 4. Vowel data.

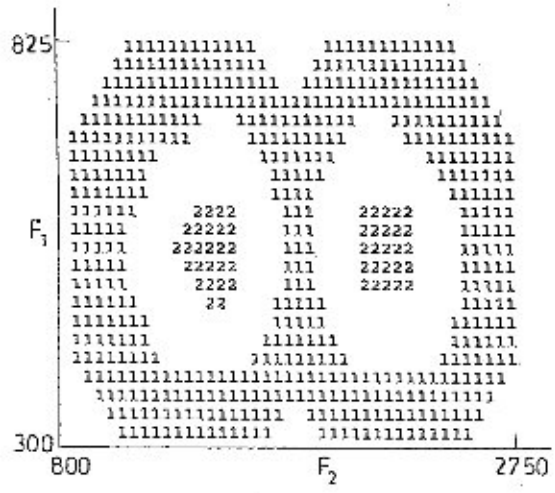


Fig. 5. Synthetic data **Pat**.

TABLE I
ATTRIBUTE-VALUE TABLE (VOWEL)

	L_1	M_1	H_1	L_2	M_2	H_2	L_3	M_3	H_3	D
x_1	0	1	0	1	1	0	1	0	0	c_1
x_2	0	0	1	1	0	0	1	1	0	c_2
x_3	1	0	0	0	0	1	0	0	1	c_3
x_4	1	0	0	1	0	0	0	1	0	c_4
x_5	1	1	0	0	0	1	0	1	0	c_5
x_6	1	1	0	1	0	0	1	0	0	c_6

otherwise. The resulting information is represented in the form of a decision table $\mathcal{S} = < U, A >$ as in Table I.

Let us explain this transformation by an example. Let a sample pattern from class c_1 have numerical components $F_1 = 600$, $F_2 = 1500$, $F_3 = 1200$. This is mapped to the nine-dimensional linguistic space with components $L_1 = 0.4$, $M_1 = 0.85$, $H_1 = 0.7$, $L_2 = 0.8$, $M_2 = 0.9$, $H_2 = 0.4$, $L_3 = 0.82$, $M_3 = 0.7$, $H_3 = 0.4$. Application of Th yields a nine-dimensional vector (0, 1, 0, 1, 1, 0, 1, 0, 0). Let class c_1 consist of n_1 pattern vectors. Each of them is transformed to this nine-dimensional form with binary components. We select the most representative template, i.e., the one with the maximum number of occurrences, from this set of n_1 templates to serve as object x_1.

U consists of six objects $x_1, \cdots, x_6$, the condition attributes are L_1, L_2, L_3, M_1, M_2, M_3, H_1, H_2, H_3 and the decision attribute set D consists of the six vowel classes $c_1, \cdots, c_6$. Each entry in row j, column i corresponds to the input $y_i^{(0)}$ for class c_j. Note that these inputs are used only for the knowledge encoding procedure. During the refinement phase, the network learns from the original $3n$-dimensional training set with $0 \leq y_i^{(0)} \leq 1$ (2).

The decision table is abbreviated by putting all the decision attributes in one column [this does not result in any ambiguity, as we assume that object x_i corresponds to the decision attribute c_i only $(i = 1, \cdots, 6)$].

The D-reducts obtained are as follows:

$(L_1 \wedge M_1 \wedge L_2)$, $(L_1 \wedge L_2 \wedge M_3)$, $(L_1 \wedge M_1 \wedge H_2)$
$(L_1 \wedge H_2 \wedge M_3)$, $(L_1 \wedge M_1 \wedge M_3)$, $(L_1 \wedge M_1 \wedge L_3 \wedge H_3)$
$(M_1 \wedge H_1 \wedge L_2 \wedge M_2)$, $(H_1 \wedge L_2 \wedge M_2 \wedge M_3)$
$(M_1 \wedge H_1 \wedge M_2 \wedge H_2)$, $(H_1 \wedge M_2 \wedge H_2 \wedge M_3)$
$(M_1 \wedge H_1 \wedge M_2 \wedge M_3)$, $(M_1 \wedge L_2 \wedge M_2 \wedge L_3)$
$(L_2 \wedge M_2 \wedge L_3 \wedge M_3)$, $(M_1 \wedge M_2 \wedge H_2 \wedge L_3)$
$(M_2 \wedge H_2 \wedge L_3 \wedge M_3)$, $(M_1 \wedge M_2 \wedge L_3 \wedge M_3)$
$(M_1 \wedge M_2 \wedge L_3 \wedge H_3)$, $(L_1 \wedge H_1 \wedge L_2 \wedge L_3 \wedge H_3)$
$(L_1 \wedge L_2 \wedge M_2 \wedge L_3 \wedge H_3)$, $(L_1 \wedge H_1 \wedge H_2 \wedge L_3 \wedge H_3)$
$(L_1 \wedge M_2 \wedge H_2 \wedge L_3 \wedge H_3)$, $(H_1 \wedge L_2 \wedge M_2 \wedge L_3 \wedge H_3)$
$(H_1 \wedge M_2 \wedge H_2 \wedge L_3 \wedge H_3)$, $(M_1 \wedge H_1 \wedge M_2 \wedge L_3 \wedge H_3)$.

Let us consider the reduct set $B = (L_1 \wedge M_1 \wedge M_3)$. Then the discernibility function $f_D^{x_i}$ (in CNF) for $i = 1, \cdots, 6$,

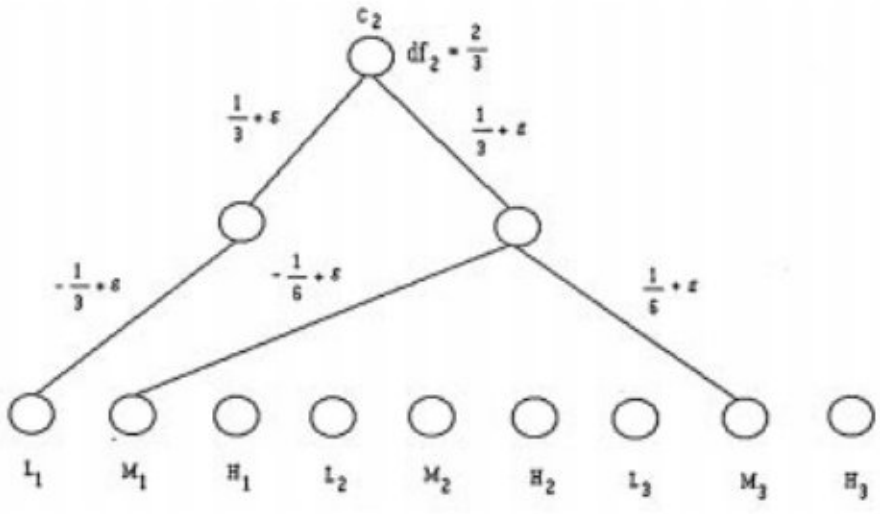


Fig. 6. Initial weight encoding for the class c_2 by Method I. remaining weights are initialized to small random values.

TABLE II
RECOGNITION SCORES (%) FOR VOWEL

Attribs.		Bayes' classifier	Q U E S T	Fuzzy MLP	Rough-Fuzzy MLP							
					R_5			R_4			R_3	
					H_1, L_2, M_2, L_3, H_3	L_1, M_2, H_2, L_3, H_3	L_1, H_1, L_2, L_3, H_3	M_1, M_2, L_3, H_3	H_1, M_2, H_2, M_3	M_1, H_1, L_2, M_2	L_1, M_1, L_2	L_1, M_1, H_2
# links		-	-	90	16	16	16	18	18	18	20	20
# inputs		-	-	9	5	5	5	4	4	4	3	3
Training		-	-	80.88	80.65	83.18	79.96	83.41	82.26	79.5	80.65	83.87
T	∂	41.6	80.6	21.6	24.3	62.2	27.0	45.9	54.1	29.7	27.0	40.5
e	a	91.1	73.3	82.2	88.9	82.2	88.9	84.4	86.7	88.9	88.9	84.4
s	i	93.0	60.5	94.1	94.1	82.4	95.3	94.1	85.9	94.1	82.4	84.7
t	u	94.7	76.3	87.8	90.2	87.8	87.8	87.8	87.8	87.8	87.8	90.2
s	e	71.1	84.6	88.7	86.8	90.6	87.7	90.6	94.3	90.5	97.2	96.2
e	o	71.1	86.7	95.1	93.9	95.1	95.1	93.9	95.1	93.9	95.1	93.9
t	Net	79.2	77.4	84.44	85.12	86.04	85.58	86.96	87.19	85.81	85.35	86.5

obtained from the discernibility matrix $\mathbf{M}_D(B)$ [using (11) and (12)] are

$$f_D^{x_1} = L_1 \wedge (M_2 \vee M_3),\ f_D^{x_2} = L_1 \wedge (M_1 \vee M_3)$$
$$f_D^{x_3} = M_1 \wedge M_3, \qquad f_D^{x_4} = L_1 \wedge M_1 \wedge M_3$$
$$f_D^{x_5} = M_1 \wedge M_3, \qquad f_D^{x_6} = L_1 \wedge M_1 \wedge M_3$$

The dependency factors df_i for the resulting rules r_i, $i = 1, \cdots, 6$ are $2/3$, $2/3$, 1, 1, 1, 1, using (13).

In the same way we consider the remaining D-reducts and find the corresponding rules and their dependency factors. These factors are encoded as the initial connection weights of the fuzzy MLP. Let us now explain the process by an example. Consider the rule r_2, *viz.* $L_1 \wedge (M_1 \vee M_3) \rightarrow c_2$ with dependency factor $df_2 = 2/3$. Here we require two hidden nodes corresponding to class c_2 to model the operator $\wedge$. The two links from the output node representing class c_2 to these two hidden nodes are assigned weights of $(df_2)/2$ to keep the weights equally distributed. From Table I we find that the entries for L_1, M_1, M_3 in case of class c_2 are 0, 0, 1, respectively. The attributes M_1 and M_3, connected by the operator $\vee$, are combined at one hidden node with link weights of $-(df_2)/4$, $(df_2)/4$, respectively, while the link weight for attribute L_1 is clamped to $-(df_2)/2$ (since there is no further bifurcation). The resultant network is finally refined during training using a training set. The performance of the network is tested on the remaining test set. Fig. 6 illustrates the weight encoding procedure for class c_2.

Table II shows the results obtained with a three-layered knowledge-based network whose connection weights are initially encoded as explained earlier. It is observed that this method works more efficiently with a smaller network. Therefore, we demonstrate the results corresponding to six hidden nodes (the lower bound in this case) only. The performance (at the end of 150 sweeps) was compared with those of a conventional MLP and a fuzzy MLP [8], having the same number of hidden nodes but with no initial knowledge encoding. It was seen that the conventional MLP with six hidden nodes is unable to classify the data. Hence this is not included in the table. The performance of the Bayes' classifier for multivariate normal patterns, using different covariance matrices are for each pattern class, is demonstrated. The choice of normal densities for the vowel data has been found to be justified [21]. The performance of the package Quest [17], implementing classification and regression trees [22], is also provided. The rough-fuzzy MLP is observed to generalize better than all the models for the test set, considering the overall scores (Net). It may also be noticed that this method generated D-reducts of different sizes. In the table, $R_n, n = 5, 4, 3$ indicates a collection of D-reducts with n components (attributes).

TABLE III
ATTRIBUTE-VALUE TABLE FOR CLASS C_6 (VOWEL)

	L_1	M_1	H_1	L_2	M_2	H_2	L_3	M_3	H_3
$x_1 - x_{20}$	1	0	0	1	0	0	1	0	0
$x_{21} - x_{29}$	1	1	0	1	0	0	1	0	0
$x_{30} - x_{36}$	1	1	0	1	0	0	0	0	1
$x_{37} - x_{41}$	1	1	0	1	0	0	0	1	0
$x_{42} - x_{45}$	1	1	0	1	0	0	0	1	1
$x_{46} - x_{49}$	1	0	0	1	0	0	0	1	1
$x_{50} - x_{51}$	0	1	0	1	0	0	0	1	1
$x_{52} - x_{53}$	0	1	0	1	0	0	1	0	0
x_{54}	1	0	0	1	0	0	1	1	0
x_{55}	0	1	0	1	0	0	0	0	1
x_{56}	1	0	0	1	0	0	0	0	1

TABLE IV
REDUCED ATTRIBUTE-VALUE TABLE FOR CLASS C_6 (VOWEL)

	L_1	M_1	H_1	L_2	M_2	H_2	L_3	M_3	H_3
$x_1 - x_{20} : (y_1)$	1	0	0	1	0	0	1	0	0
$x_{21} - x_{29} : (y_2)$	1	1	0	1	0	0	1	0	0
$x_{30} - x_{36} : (y_3)$	1	1	0	1	0	0	0	0	1
$x_{37} - x_{41} : (y_4)$	1	1	0	1	0	0	0	1	0

B. Method II

This method is applied to both the data sets Vowel and Pat. A threshold of $Th = 0.8$ was used for the Vowel data. It can be observed from Fig. 5 that the synthetic data set is uniformly distributed over the entire feature space. Therefore, setting a threshold greater than 0.5 caused problems here, such that for certain objects all three input components corresponding to a feature became clamped at zero. To circumvent this, we set Th at 0.5 for Pat.

1) Vowel Data: Each class had a separate attribute value table consisting of multiple objects. Let us consider class c_6 as an example. The first column of Table III corresponds to the objects which have the attribute-values indicated in the respective rows. We observe that the rows correspond to 20, 9, 7, 5, 4, 4, 2, 2, 1, 1, 1 objects, respectively.

After applying the threshold Tr of (14), objects $x_{42}-x_{56}$ are eliminated from the table. Hence the reduced attribute-value table (Table IV) now consists of four rows only.

The discernibility matrix for class c_6 is

	y_1	y_2	y_3	y_4
y_1	ϕ			
y_2	$\{M_1\}$	ϕ		
y_3	$\{M_1, L_3, H_3\}$	$\{L_3, H_3\}$	ϕ	
y_4	$\{M_1, L_3, M_3\}$	$\{L_3, M_3\}$	$\{M_3, H_3\}$	ϕ.

The discernibility function f for c_6 is

$$M_1 \wedge (M_1 \vee L_3 \vee H_3) \wedge (L_3 \vee H_3) \wedge (M_1 \vee L_3 \vee M_3) \wedge (L_3 \vee M_3) \wedge (M_3 \vee H_3) = M_1 \wedge (L_3 \vee H_3) \wedge (L_3 \vee M_3) \wedge (M_3 \vee H_3).$$

The disjunctive normal form of f is

$$(M_1 \wedge L_3 \wedge M_3) \vee (M_1 \wedge L_3 \wedge H_3) \vee (M_1 \wedge M_3 \wedge H_3) \vee (M_1 \wedge L_3 \wedge M_3 \wedge H_3).$$

The resultant reducts are

$$M_1 \wedge L_3 \wedge M_3,\ M_1 \wedge L_3 \wedge H_3,\ M_1 \wedge M_3 \wedge H_3,\ M_1 \wedge L_3 \wedge M_3 \wedge H_3.$$

The reduced attribute value table for reduct $M_1 \wedge L_3 \wedge M_3$ is

	M_1	L_3	M_3
y_1	0	1	0
y_2	1	1	0
y_3	1	0	0
y_4	1	0	1

The reduced discernibility matrix for $M_1 \wedge L_3 \wedge M_3$ is

	y_1	y_2	y_3	y_4
y_1	ϕ			
y_2	$\{M_1\}$	ϕ		
y_3	$\{M_1, L_3\}$	$\{L_3\}$	ϕ	
y_4	$\{M_1, L_3, M_3\}$	$\{L_3, M_3\}$	$\{M_3\}$	ϕ.

The discernibility functions f_{y_i} for each object y_i, $i = 1, 2, 3, 4$ are

$$f_{y_1} = M_1 \wedge (M_1 \vee L_3) \wedge (M_1 \vee L_3 \vee M_3) = M_1$$
$$f_{y_2} = M_1 \wedge L_3 \wedge (L_3 \vee M_3) = M_1 \wedge L_3$$
$$f_{y_3} = (M_1 \vee L_3) \wedge L_3 \wedge M_3 = M_3 \wedge L_3$$
$$f_{y_4} = (M_1 \vee L_3 \vee M_3) \wedge (L_3 \vee M_3) \wedge M_3 = M_3.$$

A dependency rule thus generated for class c_6 is

$$M_1 \vee (M_1 \wedge L_3) \vee (M_3 \wedge L_3) \vee M_3 \rightarrow c_6$$
$$\text{i.e., } M_1 \vee M_3 \rightarrow c_6.$$

The other rules for c_6 are

$$M_1 \vee H_3 \rightarrow c_6$$
$$M_1 \vee M_3 \vee H_3 \rightarrow c_6.$$

Similarly, we obtain 1, 2, 1, 1, 2 dependency rules for classes c_1, c_2, c_3, c_4, c_5, respectively. The dependency factor of each rule is one. So, considering all possible combinations we generate 12 sets of rules for the six classes. This leads to 12 possible network encodings.

A sample set of dependency rules generated for the six classes is

$$H_1 \wedge L_2 \wedge L_3 \rightarrow c_1, \quad M_1 \vee L_3 \rightarrow c_2,$$
$$M_3 \vee H_3 \rightarrow c_3, \quad M_3 \vee H_3 \rightarrow c_4$$
$$M_3 \rightarrow c_5, \quad M_1 \vee M_3 \rightarrow c_6.$$

This corresponds to the network represented in column 1 (of rough-fuzzy MLP) in Table V.

To encode the rule for class c_6 we require one hidden node for modeling the conjunct. The corresponding output node is connected to the hidden node with initial link weight of

TABLE V
RECOGNITION SCORES (%) FOR VOWEL

Attributes for		Rough-Fuzzy MLP					
c_1, c_3, c_4		$H_1 \wedge L_2 \wedge L_3$; $M_3 \vee H_3$;			$M_3 \vee H_3$		
c_2		$M_1 \vee L_3$		$M_1 \vee M_3$			
c_5		M_3				H_3	
c_6		$M_1 \vee M_3$	$M_1 \vee M_3 \vee H_3$	$M_1 \vee M_3$	$M_1 \vee H_3$	$M_1 \vee M_3$	$M_1 \vee M_3 \vee H_3$
# links		38	19	38	18	18	19
Training		85.48	80.65	81.11	80.19	83.16	82.72
T	δ	51.4	21.6	56.8	43.2	54.1	59.5
e	a	84.4	88.9	82.2	86.9	82.2	75.6
s	i	94.1	85.9	95.3	85.9	94.1	87.1
t	u	90.2	86.6	87.8	87.8	90.2	87.8
s	e	84.0	97.2	78.3	93.4	82.1	84.9
e	o	93.9	93.9	95.1	93.9	93.9	93.9
t	Net	86.27	85.13	85.13	86.27	85.81	84.44

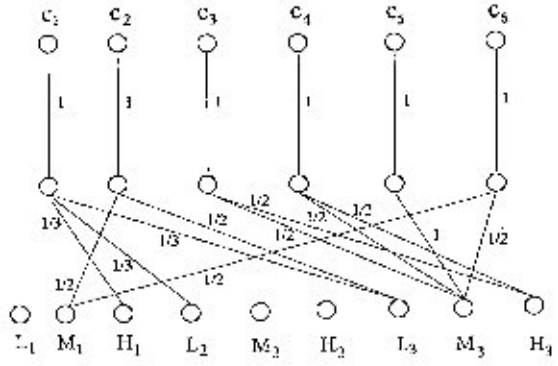


Fig. 7. Initial weights encoding by Method II. remaining weights are initialized to small random values.

TABLE VI
ATTRIBUTE-VALUE TABLE FOR CLASS C_2(PAT)

	L_1	M_1	H_1	L_2	M_2	H_2
$x_1 - x_{16}$	1	1	0	1	1	0
$x_{17} - x_{28}$	1	1	0	0	1	1
$x_{29} - x_{37}$	0	1	1	1	1	0
$x_{38} - x_{45}$	0	1	1	0	1	1
$x_{46} - x_{48}$	1	1	0	0	0	1
$x_{49} - x_{50}$	0	1	1	0	0	1
x_{51}	1	1	0	1	0	0

$df_6 = 1$. Then the input attribute pair (M_1, M_3) is connected to this hidden node with link weights $(df_6)/2(= 0.5)$. A sample network is illustrated in Fig. 7.

Table V demonstrates sample results obtained by the three-layered knowledge-based network, at the end of 150 sweeps. Unlike Method I, in all the cases Method II constructed a network with six hidden nodes and six input nodes. Its performance improves on that of the fuzzy and conventional versions of the MLP, Bayes' classifier, Quest and Method I (as observed from Table II).

2) Synthetic Data: The attribute-value table for class c_2 is depicted in Table VI. The rows correspond to 16, 12, 9, 8, 3, 2, 1 objects, respectively. Application of (14) results in the elimination of objects $x_{46} - x_{51}$. The D-reducts generated are $L_1 \wedge L_2$, $H_1 \wedge L_2$, $H_1 \wedge H_2$, $L_1 \wedge H_2$. We obtain four D-reducts for each of the other two classes. Considering all possible combinations, we generate 64 sets of rules for the three classes. This results in 64 possible network encodings.

A sample set of dependency rules for the three classes is

$$(L_1 \wedge M_2 \wedge H_2) \vee (L_1 \wedge M_1 \wedge H_2) \rightarrow c_1,\ H_1 \wedge H_2 \rightarrow c_2$$

and

$$(M_1 \wedge H_1) \vee (H_1 \wedge M_2 \wedge H_2) \vee (M_1 \wedge M_2 \wedge H_2) \rightarrow c_3.$$

This corresponds to column 2 of Table VII.

The subnetwork for class c_3 consists of three hidden nodes, each with initial output link weight of $(df_3)/3(= 0.33)$. The input attribute pair (M_1, H_1) is connected to the first of these hidden nodes with link weights $(df_3)/6(= 0.17)$. The remaining attributes (H_1, M_2, H_2) and (M_1, M_2, H_2) are connected to the next two hidden nodes with link weights $(df_3)/9(= 0.11)$.

Table VII provides a sample set of results obtained by a three-layered knowledge-based network. Note that we have simulated all 64 networks. In all cases the algorithm generated six hidden nodes. The performance was compared with that of a conventional and fuzzy MLP (all at the end of 1000 sweeps), k-NN classifier and Quest [17]. The conventional MLP failed to recognize class c_2 (e.g., the scores for classes c_1, c_2, and c_3 are 87.1, 0.0, 51.6, respectively, for the test set). The rough-fuzzy MLP generalizes better than the fuzzy MLP (with one hidden layer having six hidden nodes) for the test patterns considering the overall scores (Net). However, the k-NN classifier and Quest (classification and regression tree) are found to provide better performance. Note that the k-NN classifier is reputed to be able to generate piecewise linear decision boundaries and is quite efficient in handling concave and linearly nonseparable pattern classes. It may also be mentioned that the fuzzy MLP with more hidden layers/nodes provides results better than that of the k-NN [23]. However, we are restricted here in using six hidden nodes for maintaining parity with the rough-fuzzy MLP. It is revealed under investigation that the method of knowledge extraction using rough sets can lead to over-reduction for the data shown in Fig. 5.

TABLE VII
RECOGNITION SCORES (%) FOR PAT

		k-NN classifier		Q U E S T	F U Z Z Y MLP	Rough-Fuzzy MLP			
c_1						$(L_1 \wedge M_2 \wedge H_2) \vee (L_1 \wedge M_1 \wedge H_2)$	$(H_1 \wedge M_2 \wedge H_2) \vee (M_1 \wedge M_2 \wedge H_2)$		$(H_1 \wedge L_2 \wedge M_2) \vee (M_1 \wedge H_1 \wedge L_2)$
c_2						$H_1 \wedge H_2$		$L_1 \wedge H_2$	$H_1 \wedge H_2$
c_3		$k=$ 1	3			$(M_1 \wedge H_1) \vee (H_1 \wedge M_2 \wedge H_2) \vee (M_1 \wedge M_2 \wedge H_2)$	$(L_1 \wedge M_1) \vee (L_1 \wedge M_2 \wedge H_2) \vee (M_1 \wedge M_2 \wedge H_2)$	$(L_1 \wedge M_1) \vee (L_1 \wedge L_2 \wedge M_2) \vee (M_1 \wedge L_2 \wedge M_2)$	$(M_1 \wedge H_1) \vee (H_1 \wedge L_2 \wedge M_2) \vee (M_1 \wedge L_2 \wedge M_2)$
# links		-	-	-	54	22	22	22	22
# inpts		-	-	-	6	5	5	6	5
Train		-	-	-	52.0	84.05	77.68	80.87	83.83
T	c_1	89.6	94.2	86.5	88.08	93.08	81.92	84.23	91.54
e	c_2	88.5	88.5	74.5	69.23	69.23	61.54	65.38	57.69
s	c_3	81.9	87.7	83.5	68.39	68.39	80.65	81.94	67.1
t	Net	86.8	91.6	84.8	80.05	82.99	80.27	82.31	80.95

Remarks:

1) Method I is based on the assumption that there is one decision attribute corresponding to each object, i.e., the classes are considered to be convex with single representative points. This method is not a special case of Method II, though the latter deals with multiple representative points for each class. For example, in Method I we simultaneously generated six rules corresponding to six vowel clauses from the same attribute-value table. On the other hand, Method II involves separate attribute-value tables for each of the six vowel classes. Therefore, a rule corresponding to one class is generated at a time from one such table. This cannot be boiled down to Method I as a special case.
2) We have transformed the decision table constructed from the initial data by dividing it into subtables, each corresponding to a decision attribute of the given system. The initial table gave rise to discernibility functions [computed by (12)] with too large a number of components and hence, a network with a huge number of hidden nodes. The computational complexity of such a network was not considered to be feasible. On the contrary, the subtables resulted in the generation of discernibility functions with less components and thus finally, a less cumbersome (more efficient) network.
3) Each decision table considered so far is clearly consistent.
4) Any comparative study of the performance of our model should consider the fact that here the appropriate number of hidden nodes is automatically generated by the rough-set theoretic knowledge encoding procedure. On the other hand, both the fuzzy and conventional versions of the MLP are required to empirically generate a suitable size of the hidden layer(s). Hence, this can be considered to be an added advantage.

VI. CONCLUSIONS

A methodology integrating rough sets with fuzzy MLP for designing a knowledge-based network is presented. The effectiveness of rough set theory is utilized for encoding the crude domain knowledge through concepts like discernibility matrix and function, reducts, and dependency factors. Two algorithms, applicable to convex and concave decision regions, are derived. This investigation not only demonstrates a way of integrating rough sets with neural networks and fuzzy sets, but also provides methods that are capable of generating the appropriate network architecture and improving the classification performance. The incorporation of fuzziness at various levels of fuzzy MLP also helps the resulting knowledge-based system to efficiently handle uncertain and ambiguous information both at the input and the output.

As was remarked earlier, a study of an integration, involving only neural nets and rough sets, was presented by Yasdi [15]. However, only one layer of adaptive weights was considered while the input and output layers involved fixed binary weights. Max, Min and Or operators were applied at the hidden nodes. Besides, the model was not tested on any real problem and no comparative study was provided to bring out the effectiveness of this hybrid approach. We, on the other hand, consider here an integration of the three paradigms, *viz.*, neural nets, rough sets and fuzzy sets. The process of rule generation and mapping of the dependency factors to the connection weight values is novel to our approach. Moreover, the three-layered MLP used has adaptive weights at all layers. These are initially encoded with the knowledge extracted from the data domain in the form of dependency rules, and later refined by training. Effectiveness of the model is demonstrated on both real-life and artificial data.

Our objective was to demonstrate the effectiveness of the rough-fuzzy MLP on *difficult* classification problems. The data set used involves the overlapping classes of the **Vowel** data and the linearly nonseparable, nonconvex, disjoint classes of the **Pat** data. Both cases could not be suitably classified by the conventional MLP. The fuzzy MLP splits the feature space into 3^n overlapping linguistic partitions, thereby handling more local information about the input. The output is modeled in terms of class membership values, appropriately taking care of fuzzy/overlapping classes. This accounts for the suitability of the fuzzy MLP in classifying these data. Incorporation of rough set-theoretic concepts for encoding the initial knowledge of the fuzzy MLP enabled the generation of the appropriate network topology using nonempirical means. Certain benchmark problems like the classification of Fisher's Iris data [24] have also been attempted. As the conventional MLP was sufficiently accurate in classifying this data, there was no noticeable improvement in the network performance

by incorporating the more complicated rough-fuzzy concept. This suggests that the rough-fuzzy MLP can be effectively used for handling cases where the conventional MLP fails.

Rough set-theoretic techniques are easily applicable to attribute-value tables with binary entries. This encouraged us to transform the continuous-valued data to this form. However, we are currently engaged in extending the algorithm to work directly on real numbers lying in [0, 1]. This forms the next part of our research.

There are several other related approaches for classification, other than neural networks. These include the ID3 algorithm [25] and classification and regression trees [22], [17]. ID3 can be very effective under certain conditions, specially if the data consists of nonnumeric feature values [25]. Numeric data needs to be optimally quantized to become applicable. This is not a trivial problem. Application of ID3 to the $3n$-dimensional linguistic feature space is an interesting alternative, to the neuro-rough approach, for future investigation. Handling of *noisy* classification problems, where the distributions of observations from the different classes overlap, is difficult using the classification and regression trees [26]. This is evident from Table II. Another interesting direction of future research would be to incorporate the fuzzy membership concept in such tree structures, to circumvent this problem.

ACKNOWLEDGMENT

The authors thank the anonymous referees for their helpful criticism.

REFERENCES

[1] *Proc. 3rd Wkshp. Rough Sets Soft Comput. (RSSC'94)*, San Jose, CA, Nov. 1994.

[2] *Proc. 4th Int. Conf.n Soft Comput. (IIZUKA96)*, Iizulka, Fukuoka, Japan, Oct. 1996.

[3] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Commun. ACM*, vol. 37, pp. 77–84, 1994.

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

[5] *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Sept. 1996.

[6] L. A. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338–353, 1965.

[7] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 4, pp. 4–22, 1987.

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

[9] S. Mitra and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, 1995.

[10] L. M. Fu, "Knowledge-based connectionism for revising domain theories," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 173–182, 1993.

[11] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial Intell.*, vol. 70, pp. 119–165, 1994.

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

[13] A. Skowron and C. Rauszer, "The discernibility matrices and functions in information systems," in *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, R. Slowinski, Ed. Dordrecht, The Netherlands: Kluwer, 1992, pp. 331–362.

[14] R. Slowinski, Ed., *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*. Dordrecht: Kluwer, 1992.

[15] R. Yasdi, "Combining rough sets learning and neural learning method to deal with uncertain and imprecise information," *Neuro-Computing*, vol. 7, pp. 61–84, 1995.

[16] A. Czyzewski and A. Kaczmarek, "Speech recognition systems based on rough sets and neural networks," in *Proc. 3rd Wkshp. Rough Sets and Soft Computing (RSSC'94)*, San Jose, CA, 1994, pp. 97–100.

[17] W. Y. Loh and Y. S. Shih, "Quest user guide," Dept. Statist., Univ. Wisconsin, Madison, Tech. Rep., 1996.

[18] S. K. Pal and D. P. Mandal, "Linguistic recognition system based on approximate reasoning," *Inform. Sci.*, vol. 61, pp. 135–161, 1992.

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

[20] ——, "Fuzzy sets and decision making approaches in vowel and speaker recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 7, pp. 623–629, 1977.

[21] S. K. Pal, "Studies on the application of fuzzy set-theoretic approach in some problems of pattern recognition and man-machine communication by voice," Ph.D. dissertation, Univ. Calcutta, India, 1978.

[22] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks/Cole, 1984.

[23] S. K. Pal and S. Mitra, "Fuzzy versions of Kohonen's net and MLP-based classification: Performance evaluation for certain nonconvex decision regions," *Inform. Sci.*, vol. 76, pp. 297–337, 1994.

[24] R. A. Fisher, "The use of multiple measurements in taxonomic problem," *Ann. Eugenics*, vol. 7, pp. 179–188, 1936.

[25] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[26] B. D. Ripley, *Pattern Recognition and Neural Networks*. New York: Cambridge Univ. Press, 1996.

Mohua Banerjee was born in 1965. She received the Ph.D. degree in pure mathematics from the University of Calcutta, India, in 1995.

From 1995 to 1997, she was a Research Associate at the Machine Intelligence Unit, Indian Statistical Institute, Calcutta. Currently, she is working as a Lecturer in the Department of Mathematics, Indian Institute of Technology, Kanpur. Her main interests include pure and applied logics, rough set theory, and its applications to approximate reasoning, fuzzy set theory, and category theory. She has had the occasion to work with distinguished scientists including Z. Pawlak (the proposer of rough set theory) while on academic visits to Poland. She has lectured on rough sets and interacted with researchers in various institutes in India and abroad.

Dr. Banerjee was awarded the Indian National Science Academy Medal for Young Scientists in 1995.

Sushmita Mitra received the B.Sc. (Hons.) degree in physics and the B.Tech. and M.Tech. degrees in computer science from the University of Calcutta in 1984, 1987, and 1989, respectively. She received the Ph.D. degree in computer science by the Indian Statistical Institute, Calcutta, in 1995.

Since 1995, she has been an Associate Professor of the Indian Statistical Institute, Calcutta, which she joined in 1989. From 1978 to 1983, she was a recipient of the National Talent Search Scholarship from the National Council for Educational Research and Training, India. From 1992 to 1994, she was in the European Laboratory for Intelligent Techniques Engineering, Aachen, as a German Academic Exchange Service (DAAD) Fellowship holder. Her research interests include pattern recognition, fuzzy sets, artificial intelligence, neural networks, and soft computing.

Dr. Mitra was awarded the 1994 IEEE TNN Outstanding Paper Award for a paper published in the IEEE TRANSACTIONS ON NEURAL NETWORKS.

Sankar K. Pal (M'81–SM'84–F'93) received the M.Tech. and Ph.D. degrees in radiophysics and electronics from the University of Calcutta, India, in 1974 and 1979, respectively. In 1982, he received another Ph.D. degree in electrical engineering along with DIC from Imperial College, University of London.

In 1986, he was awarded a Fulbright Postdoctoral Visiting Fellowship to work at the University of California, Berkeley, and the University of Maryland, College Park. In 1989 he received an NRC-NASA Senior Research Award to work at the NASA Johnson Space Center, Houston, TX, USA. He is a Distinguished Scientist and Founding Head of Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. He was appointed a Distinguished Visitor of IEEE Computer Society for the Asia-Pacific Region. His research interests mainly include pattern recognition, image processing, neural nets, genetic algorithms, and fuzzy sets and systems. He is a coauthor of the book *Fuzzy Mathematical Approach to Pattern Recognition* (New York: Wiley, 1986) and a coeditor of three books: *Fuzzy Models for Pattern Recognition* (New York: IEEE Press, 1992), *Genetic Algorithms for Pattern Recognition* (Boca Raton, FL, CRC, 1996), and *Rough-Fuzzy Hybidization: New Trend in Decision Making* (New York: Springer-Verlag, 1998).

Dr. Pal is a Fellow of the Indian National Science Academy, the Indian Academy of Sciences, the National Academy of Sciences of India, and the Indian National Academy of Engineering. He received the 1990 Shanti Swarup Bhatnagar Prize in engineering sciences, the Jawaharlal Nehru Fellowship, the Vikram Sarabhai Research Award, and the NASA Tech Brief Award, all in 1993. He received the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award in 1994, the NASA Patent Application Award in 1995, and in 1997, the IETE–Ram Lal Wadhwa Gold Medal. He is a Memberof the Executive Advisory Editorial Board for the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the *International Journal of Approximate Reasoning*, North Holland, and an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, *Fuzzy Sets and Systems*, *Pattern Recognition Letters*, *Neurocomputing*, *Applied Intelligence*, *Information Sciences*, *International Journal of Knowledge Based Intelligent Engineering Systems*, and *Far-East Journal of Mathematical Sciences*. He was also the Guest Editor of an *IEEE Computer* special issue on Neural Networks: Theory and Practice, March 1996, and *JIETE* special issue on Neural Networks, July–October, 1996.