Rough Knowledge-based Network, Fuzziness and Classification

S. Mitra, M. Banerjee and S. K. Pal

Machine Intelligence Unit, Indian Statistical Institute, Calcutta, India.

A method of integrating rough sets and fuzzy multilayer perceptron (MLP) for designing a knowledge-based network for pattern recognition problems is described. Rough set theory is used to extract crude knowledge from the input domain in the form of rules. The syntax of these rules automatically determines the optimal number of hidden nodes while the dependency factors are used in the initial weight encoding. Results on classification of speech data demonstrate the superiority of the system over the fuzzy and conventional versions of the MLP.

Keywords: Classification; Fuzzy MLP; Knowledge-based networks; Rough sets

1. 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]. During hybridisation, the individual tools act synergetically (not competitively) to increase the application domain of each other. The present article describes such a hybrid paradigm involving rough sets, fuzzy sets and Artificial Neural Networks (ANNs) to design a knowledge-based network for pattern recognition.

Generally, ANNs consider a fixed topology of neurons connected by links in a pre-defined manner. These connection weights are usually initialised by small random values. Knowledge-based networks [4,5] constitute a special class of ANNs 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.

The theory of rough sets [6] 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* [7], *reducts* and *dependency factors* [6], 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

Correspondence and offprint requests to: S. Mitra, Machine Intelligence Unit, Indian Statistical Institute, 203, B.T. Road, Calcutta-700035, India. E-mail: {sushmita,miux9503,sankar}@isical.ernet.in.

rough set formalism, especially in problems of machine learning from examples and control theory [8]. In the context of neural networks, an attempt of such implementation has been made by Yasdi [9]. 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 [10,11]. However, in these methods, rough sets were used either for knowledge discovery at the level of data acquisition (viz., in preprocessing of the feature vectors) [10] or for reduction of the initial data set [11], and not for structuring the network.

In this article, we have attempted to integrate rough sets and a three-layered fuzzy multilayer perceptron (MLP) for designing a knowledge-based system. The fuzzy MLP described in [12, 13] incorporates fuzzy set-theoretic concepts at the input and output levels and during learning. The input is modeled in terms of the $3n$-dimensional linguistic feature space while the output consists of class membership values. Rough set theory is utilised for extracting crude domain knowledge that is encoded among the connection weights. A method is derived to model convex decision regions with single object representatives. The feature space gives us the condition attributes and the output classes the decision attributes, so as to result in a decision table. Rules are then generated from the 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 [4,5], involves a non-binary weighting mechanism based on a detailed and systematic estimation of the available domain information. It may be noted that the optimal 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.

2. Fuzzy MLP Model

In this section we describe, in brief, the fuzzy MLP [12] which is used for designing the knowledge-based network. Consider the three-layered network

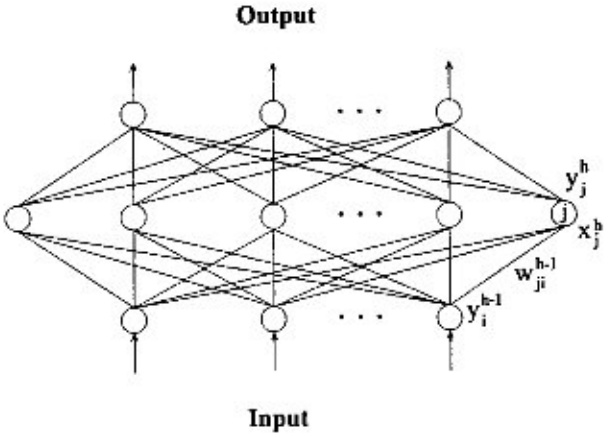

Fig. 1. The three-layered MLP.

given in Fig. 1. The output of a neuron in any layer (h) other than the input layer 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 minimised by the backpropagation algorithm using a gradient descent with a gradual decrease of the gain factor.

2.1. Input Vector

An n-dimensional pattern $\mathbf{F_i} = [F_{i1}, F_{i2}, \ldots, F_{in}]$ is represented as a $3n$-dimensional vector [14]

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

where the μ values indicate the membership functions of the corresponding linguistic π-sets *low*, *medium* and *high* along each feature axis. The three overlapping π-sets along a feature axis are depicted in Fig. 2.

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

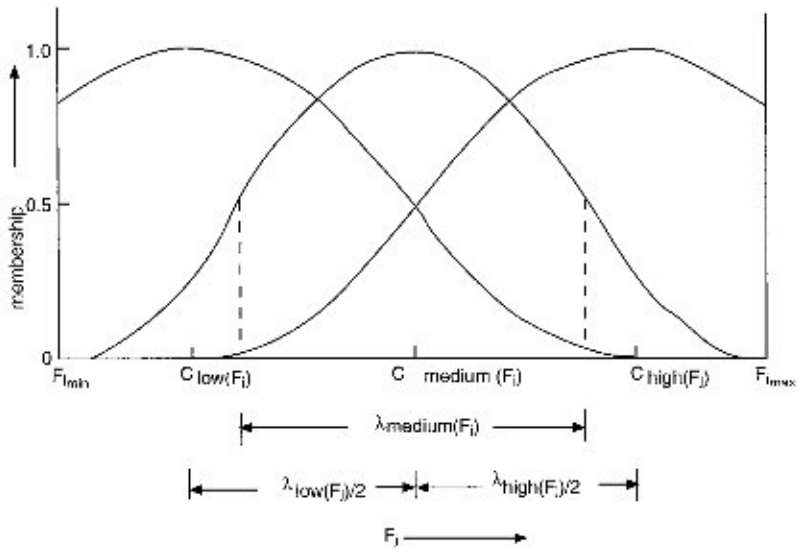

Fig. 2. Overlapping linguistic π-sets.

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

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*, *medium* and *high* are represented as [14, 12, 15]

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

$$medium = \left\{ \frac{\pi\left(F_j\left(\frac{0.95}{M}\right); c_{j_l}, \lambda_{j_l}\right)}{L}, \frac{0.95}{M}, \frac{\pi\left(F_j\left(\frac{0.95}{M}\right); c_{j_h}, \lambda_{j_h}\right)}{H} \right\}$$

$$high = \left\{ \frac{\pi\left(F_j\left(\frac{0.95}{H}\right); c_{j_l}, \lambda_{j_l}\right)}{L}, \frac{\lambda\left(F_j\left(\frac{0.95}{H}\right); c_{j_m}, \lambda_{j_m}\right)}{M}, \frac{0.95}{H} \right\}. \quad (4)$$

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

2.2. Output Representation

Consider an l-class problem domain such that we have l nodes in the output layer. The membership of the ith pattern in class k, lying in the range [0,1], is defined as [16]

$$\mu_k(\mathbf{F_i}) = \frac{1}{1 + \left(\dfrac{z_{ik}}{f_d}\right)^{f_e}} \quad (5)$$

where z_{ik} is the weighted distance of the training pattern $\mathbf{F_i}$ from class c_k, and the 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

$$d_j = \mu_j(\mathbf{F_i}) \quad (6)$$

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.

Let us consider a simple example to explain how the fuzzy MLP works. Let the two-dimensional input for the ith pattern $\mathbf{F_i}$ be given in linguistic form as F_{i1} = low, F_{i2} = medium. This translates into a six-dimensional vector with components (say) [0.95, 0.6, 0.02, 0.7, 0.95, 0.7] by Eqs (2)–(4). Let there be two output classes, such that the desired two-

dimensional output vector $\mathbf{d}$ has components [0.9, 0.3] by Eqs (5)–(6). This implies that $\mathbf{F}_i$ belongs to class 1 with membership 0.9 and to class 2 with membership 0.3.

Such input–output pairs are used for training the fuzzy MLP. During testing, when an unknown pattern is clamped at the input, the trained fuzzy MLP infers its class membership vector $\mathbf{y}^{(H)}$ by Eq. (1), where H corresponds to the output layer. This represents the degree of belongingness of the test pattern to the various classes.

3. Rough Set Preliminaries

Here we present some requisite preliminaries of rough set theory, the details of which are available in [6,7]. Before that, let us provide an example to explain the concept of rough set theory. Consider an *information system* $< U, \{a\} >$, 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 following steps:

(i) 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.
(ii) 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 $< U, \{a\} >$.

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

An *information system* is a pair $\mathcal{S} = < U, A >$, where U is a non-empty finite set called the *universe* and A a non-empty 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 labelled 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 $I_B = \cap_{a \in B} I_a$.

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.

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

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

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

$$f_{\mathcal{S}}(\bar{a}_1, \ldots, \bar{a}_m) = \wedge\{\vee(c_{ij}) : 1 \leq j < i \leq n,\ c_{ij} \neq \emptyset\}, \tag{8}$$

where $\vee(c_{ij})$ is the disjunction of all variables $\bar{a}$ with $a \in c_{ij}$.

An attribute $b \in B (\subseteq A)$ is *dispensable* in B if $I_B = I_{B \setminus \{b\}}$, otherwise b is *indispensable* in B. $B (\subseteq A)$ is *independent* in $\mathcal{S}$ if every attribute from B is indispensable in B. Otherwise, B is *dependent* in $\mathcal{S}$. B is called a *reduct* in $\mathcal{S}$ if B is independent in $\mathcal{S}$ and $I_B = I_A$.

It can be seen [7] that $\{a_{i_1}, \ldots, a_{i_p}\}$ is a reduct in $\mathcal{S}$ if and only if $a_{i_1} \wedge \ldots \wedge a_{i_p}$ is a prime implicant (constituent of the disjunctive normal form) of $f_{\mathcal{S}}$.

Let $B, C \subseteq A$. C *depends* on B if and only if $I_B \subseteq I_C$, i.e. 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, $0 \leq df \leq 1$. Formally,

$$df = \frac{\text{card}(POS_B(C))}{\text{card}(U)} \tag{9}$$

where $POS_B(C) = \cup_{X \in I_C} \underline{B}X$, and card denotes cardinality of the set.

We are concerned with a specific type of information system $\mathcal{S} = < U, A >$, 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.

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 $POS_B(D) = POS_{B\setminus\{b\}}(D)$; 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 $POS_C(D) = POS_B(D)$.

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

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

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

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

4. Network Configuration using Rough Sets

Here we formulate a method for rule generation and knowledge encoding for configuring an optimal network. It works on the assumption that each object of the domain of discourse corresponds to a single decision attribute. From the perspective of pattern recognition, this implies using a single prototype to model a (convex) 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 minimises 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 rule generation and knowledge encoding procedure.

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.

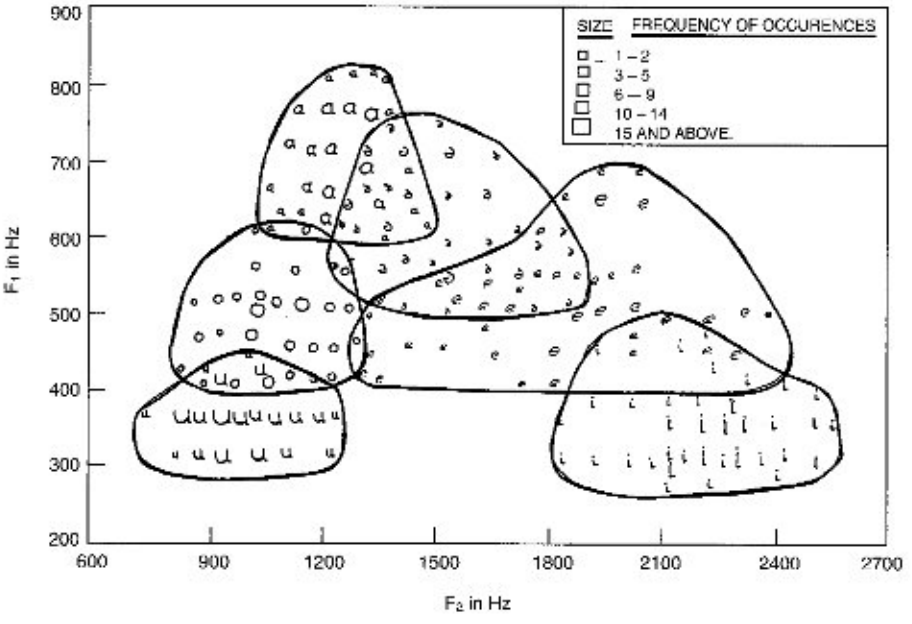

Fig. 3. Block diagram of the rule generation and knowledge encoding procedure.

4.1. Rule Generation

For each D-reduct $B = \{b_1, \ldots, b_k\}$ (say), we define a discernibility matrix (denoted $\mathbf{M}_D(B)$) from the D-discernibility matrix (given by Eq. (10)) as follows:

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

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

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

$$f_D^{x_i} = \wedge\{\vee(c_{ij}) : 1 \leq j \leq n, j \neq i, c_{ij} \neq \phi\} \quad (12)$$

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

$f_D^{x_i}$ is brought to its conjunctive normal form (c.n.f.) P_i. For $i = 1, \ldots, n$, $f_D^{x_i}$ 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 c.n.f. is a disjunct of attributes, say $b_{i_1}, \ldots, b_{i_p} \in B$, we consider the transitive closure of the union of the relations $I_{b_{i_1}}, \ldots, I_{b_{i_p}}$. Let I_i denote the intersection of all these equivalence relations.

The dependency factor df_i for r_i is then given by

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

where $POS_i(d_i) = \cup_{X \in I_{d_i}} l_i(X)$, and $l_i(X)$ is the lower approximation of X with respect to I_i.

4.2. Knowledge Encoding

Here, we formulate a methodology for encoding initial knowledge in the fuzzy MLP of ref. [12], 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 3-dimensional feature space of $\mu_{low(F_{ij})}(\mathbf{F_i})$, $\mu_{medium(F_{ij})}(\mathbf{F_i})$ and $\mu_{high(F_{ij})}(\mathbf{F_i})$ by Eq. (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 1, while the others are fixed at 0. 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 modelled 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 Eq. (13). The weight $w_{ik}^{(2)}$ between a hidden node i and output node k is set at $\alpha/fac + \epsilon$, 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 $w_{a_j i}^{(1)}$ 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.

5. Implementation and Experimental Results

Here we implement the method on real life speech data. The initial weight encoding scheme is demonstrated and recognition scores are presented.

The speech data **Vowel** [17] deals with 871

samples of the six Indian Telugu vowel sounds (∂,*a*,*i*,*u*,*e*,*o*). These were uttered in a Consonant-Vowel-Consonant context by three male speakers in the age group of 30 to 35 years. The overlapping data set shown in Fig. 4 (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. Let the above classes be denoted as c_1, ..., c_6, respectively.

The training set considered 50 per cent of the data selected randomly from each of the pattern classes. The remaining 50 per cent 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.

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*n*-dimensional linguistic space of Eq. (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 0 otherwise. The resulting information is represented in the form of a decision table $\mathcal{S} = < U, A >$ as in Table 1. U consists of six objects x_1, ..., 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, ...,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 3*n*-dimensional training set with $0 \leq y_i^{(0)} \leq 1$ (Eq. (2)).

Table 1. 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

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, ...,6)).

The D-reducts obtained are as follows:

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

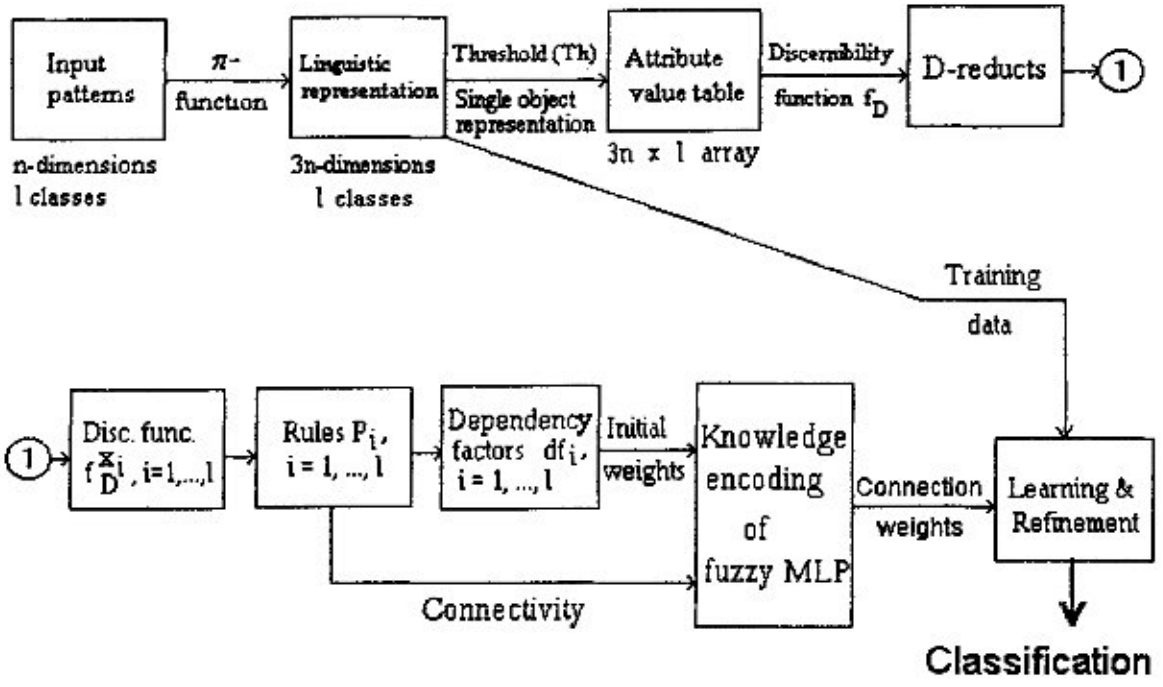

Fig. 4. Vowel data.

$\wedge$ 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 $f_D^{r_i}$ (in c.n.f.) for $i = 1, \ldots, 6$, obtained from the discernibility matrix $\mathbf{M}_D(B)$ (using Eqs (11) and (12)) are:

$f_D^{r_1} = L_1 \wedge (M_1 \vee M_3)$, $f_D^{r_2} = L_1 \wedge (M_1 \vee M_3)$, $f_D^{r_3} = M_1 \wedge M_3$, $f_D^{r_4} = L_1 \wedge M_1 \wedge M_3$, $f_D^{r_5} = M_1 \wedge M_3$, $f_D^{r_6} = L_1 \wedge M_1 \wedge M_3$.

The dependency factors df_i for the resulting rules r_i, $i = 1, \ldots, 6$ are 2/3, 2/3, 1, 1, 1, 1, using Eq. (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 1 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). All other connection weights are assigned very small random weights ϵ, lying in the range $[-0.005, +0.005]$. The resultant network is finally refined during training using a training set. The performance of the network is tested on the remaining test set. Figure 5 illustrates this weight encoding procedure for class c_2.

Table 2 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 was compared with those of a conventional MLP and a fuzzy MLP [12], 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. It may be noticed that this method generated D-reducts of different sizes. In the table, R_n indicates a collection of D-reducts with n components (attributes).

6. Conclusions

The present paper describes a methodology for integrating rough sets and fuzzy MLP for designing a knowledge-based network. Rough set theory is utilised for encoding the crude domain knowledge in the form of rules. This investigation also provides a method that is capable of generating the optimal network architecture and improving the classification performance.

As was noted earlier, a study of an integration,

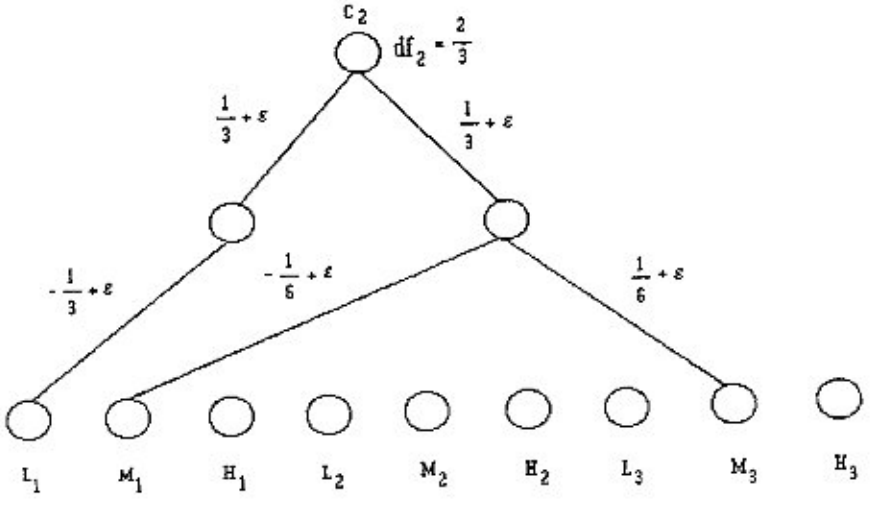

Fig. 5. Initial weight encoding for class c_2. Remaining weights are initialised to small random values.

Table 2. Recognition scores (%) for Vowel.

Attributes		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	L_1,L_2, M_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
training		80.88	80.65	83.18	79.96	84.56	83.41	82.26	79.5	80.65	83.87
T	*a*	21.6	24.3	62.2	27.0	64.9	45.9	54.1	29.7	27.0	40.5
e	*a*	82.2	88.9	82.2	88.9	77.8	84.4	86.7	88.9	88.9	84.4
s	*i*	94.1	94.1	82.4	95.3	84.7	94.1	85.9	94.1	82.4	84.7
r	*u*	87.8	90.2	87.8	87.8	87.8	87.8	87.8	87.8	87.8	90.2
s	*e*	88.7	86.8	90.6	87.7	80.2	90.6	94.3	90.5	97.2	96.2
e	*o*	95.1	93.9	95.1	95.1	96.3	93.9	95.1	93.9	95.1	93.9
t	Net	84.44	85.12	86.04	85.58	83.98	86.96	87.19	85.81	85.35	86.5

involving only neural nets and rough sets, was presented by Yasdi [9]. 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. Effectiveness of the model is demonstrated on speech data.

Acknowledgements. The research of Dr. M. Banerjee is supported by the CSIR, Government of India. This work is also sponsored by the CSIR grant no. 25 (0093)/97/EMR-II.

References

1. Proc Third Workshop on Rough Sets and Soft Computing (RSSC'94), San José, USA, November 1994.
2. Proc Fourth International Conference on Soft Computing (IIZUKA96), Iizuka, Fukuoka, Japan, October 1996.
3. Zadeh LA. Fuzzy logic, neural networks, and soft computing. Comm ACM 1994; 37: 77–84.
4. Fu LM. Knowledge-based connectionism for revising domain theories. IEEE Trans Systems, Man and Cybern 1993; 23: 173–182.
5. Towell GG, Shavlik JW. Knowledge-based artificial neural networks. Art Intell 1994; 70: 119–165.
6. Pawlak Z. Rough Sets, Theoretical Aspects of Reasoning about Data. Kluwer Academic, Dordrecht, 1991.
7. Skowron A, Rauszer C. The discernibility matrices and functions in information systems. In R. Slowiński, ed. Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic, Dordrecht, 1992, pp. 331–362.
8. Slowiński R. ed. Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic, Dordrecht, 1992.
9. Yasdi R. Combining rough sets learning and neural learning method to deal with uncertain and imprecise information. Neurocomputing 1995; 7: 61–84.
10. Czyzewski A, Kaczmarek A. Speech recognition systems based on rough sets and neural networks. Proc Third Workshop on Rough Sets and Soft Computing (RSSC'94), San José, USA, 1994, pp. 97–100.
11. Szczuka MS. Rough set methods for constructing neural networks, Vol. 7, Engineering Systems Design and Analysis, 1996, pp. 9–14.
12. Pal SK, Mitra S. Multi-layer perceptron, fuzzy sets and classification. IEEE Trans Neural Networks 1992; 3: 683–697.
13. Mitra S, Pal SK. Fuzzy multi-layer perceptron, inferencing and rule generation. IEEE Trans Neural Networks 1995; 6: 51–63.
14. Pal SK, Mandal DP. Linguistic recognition system based on approximate reasoning. Infor Sci 1992; 61: 135–161.
15. Mitra S. Fuzzy MLP based expert system for medical diagnosis. Fuzzy Sets and Systems 1994; 65: 285–296.
16. Pal SK, Dutta Majumder D. Fuzzy Mathematical Approach to Pattern Recognition. John Wiley, New York 1986.
17. Pal SK, Dutta Majumder D. Fuzzy sets and decision making approaches in vowel and speaker recognition. IEEE Trans Systems, Man and Cybern 1977; 7: 625–629.