Knowledge-Based Fuzzy MLP for Classification and Rule Generation

Sushmita Mitra, Rajat K. De, and Sankar K. Pal, *Fellow, IEEE*

***Abstract*— A new scheme of knowledge-based classification and rule generation using a fuzzy multilayer perceptron (MLP) is proposed. Knowledge collected from a data set is initially encoded among the connection weights in terms of class *a priori* probabilities. This encoding also includes incorporation of hidden nodes corresponding to both the pattern classes and their complementary regions. The network architecture, in terms of both links and nodes, is then refined during training. Node growing and link pruning are also resorted to. Rules are generated from the trained network using the input, output, and connection weights in order to justify any decision(s) reached. Negative rules corresponding to a pattern not belonging to a class can also be obtained. These are useful for inferencing in ambiguous cases. Results on real life and synthetic data demonstrate that the speed of learning and classification performance of the proposed scheme are better than that obtained with the fuzzy and conventional versions of the MLP (involving no initial knowledge encoding). Both convex and concave decision regions are considered in the process.**

***Index Terms*—Classification, fuzzy MLP, knowledge-based networks, rule generation.**

I. Introduction

KNOWLEDGE-BASED networks [1], [2] constitute a special class of artificial neural networks (ANN's) [3], [4] 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.

Connectionist expert systems [5], [6] use the set of connection weights of a *trained* neural net for encoding the knowledge base for the problem under consideration. These models are usually suitable in data-rich environment. When a fuzzy neural net constitutes the knowledge base, we call the model a neuro-fuzzy expert system [7]. This accommodates the merits of neuro-fuzzy computing, *viz.*, parallelism, fault tolerance, adaptivity, and uncertainty management, in expert system design. Recently, there have been some attempts in improving the performance of connectionist expert systems using knowledge-based networks. Such a model has the capability of outperforming a standard MLP as well as other related algorithms.

Some related works in this area include the model by Gallant [5], dealing with *sacrophagal* problems, that uses *crisp* inputs/outputs and a linear discriminant network (with no hidden nodes) trained by the simple *Pocket Algorithm*. Yin and Liang [8] incrementally built a dynamic knowledge base capable of both acquiring new knowledge as well as relearning existing information. Fu [1] used the initial domain knowledge (in terms of rules) to generate the network topology, while the links were weighted to maintain the semantics. Towell and Shavlik [2] mapped problem-specific "domain theories" into layered neural networks and then refined this reformulated knowledge using backpropagation. Machado and Rocha [9] used a connectionist knowledge base involving fuzzy numbers at the input layer, fuzzy "*and*" at the hidden layers, and fuzzy "*or*" at the output layer.

In this article we consider a new idea of knowledge encoding among the connection weights of a fuzzy MLP [10]. The methodology involves development of a technique for generating an appropriate architecture of the fuzzy MLP [10] in terms of hidden nodes and links. To demonstrate its significance an application to pattern classification has been provided, as an example. The model is capable of generating both *positive* (indicating the belongingness of a pattern to a class) and *negative* rules (indicating *not* belongingness of a pattern to a class) in linguistic form to justify any decision reached. This is found to be useful for inferencing in ambiguous cases. Note that, the rule generation procedures described in this article are different from that reported in [11]. The model is capable of handling input in numerical, linguistic, and set forms, and can tackle uncertainty due to overlapping classes. The knowledge encoding procedure, unlike most other methods [1], [2], involves a nonbinary weighting mechanism.

It is found that the classification performance improves appreciably with the encoding of the initial knowledge in the network architecture. The proposed network model converges much earlier and hence more meaningful rules are generated at this stage as compared to the other models. A brief description of the fuzzy MLP used is provided in Section II. In Section III we introduce the knowledge encoding methodology that makes it more efficient. The algorithms for rule generation are provided in Section IV. The model is implemented on synthetic, and real-life speech and medical data (in Section V) for both classification and rule generation. Its performance is also compared with that of the conventional fuzzy versions of the MLP and fuzzy min–max neural network [12]. The paper is concluded in Section VI.

II. The Fuzzy MLP Model

In this section we describe the fuzzy MLP [10] used. The output of a neuron in any layer $(h+1)$ other than the input layer is given as

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

where $y_i^{(h)}$ is the state of the ith neuron in the preceding hth layer and $w_{ji}^{(h)}$ is the weight of the connection from the ith neuron in layer (h) to the jth neuron in layer $(h+1)$. For nodes in the input layer, $y_j^{(0)}$ corresponds to the jth component of the input vector. 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 $\boldsymbol{F}_i = [F_{i1}, F_{i2}, \cdots, F_{in}]$ is represented as a $3n$-dimensional vector [13]

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

where the μ values indicate the membership functions of the corresponding linguistic π-sets [14], [10] along each feature axis. The input can be in numeric, linguistic or set form and can have modifiers *very, more or less (mol)*, or *not* attached to it as described in [13]. We ensure that any feature value along the jth axis for pattern $\boldsymbol{F}_i$ is assigned membership value combinations in the corresponding three-dimensional (3-D) linguistic space of (2) in such a way that at least one of $\mu_{\text{low}(F_{ij})}(\boldsymbol{F}_i), \mu_{\text{medium}(F_{ij})}(\boldsymbol{F}_i)$ or $\mu_{\text{high}(F_{ij})}(\boldsymbol{F}_i)$ is greater than 0.5. This heuristic ensures that each pattern point belongs positively to at least one of the linguistic sets *low, medium,* or *high* along each feature axis.

B. Output Representation

Consider an l-class problem domain such that we have l nodes in the output layer. The desired output $(d_k \in [0,1])$ of the kth output node for the ith input pattern, is defined as [14]

$$d_k = \mu_k(\boldsymbol{F}_i) = \frac{1}{1+\left(\frac{z_{ik}}{f_d}\right)^{f_e}} \tag{3}$$

where $\mu_k(\mathbf{F}_i)$ is the membership value of the ith pattern in class C_k, z_{ik} is the weighted distance of the training pattern $\boldsymbol{F}_i$ from 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. They influence the amount of overlapping among the output classes. Note that, here we have used a (nonlinguistic) definition of the output nodes which indicates the degree of belongingness of a pattern to a class. However, this definition may be suitably modified in other application areas to include linguistic definitions.

III. Knowledge-Based Classification

In this section, we formulate a methodology for encoding *a priori* initial knowledge in the fuzzy MLP. Our concept is based on the fact that if a classifier is initially provided with some knowledge from the data set, the resulting searching space is reduced thereby leading to a more efficient learning. The architecture of the network may become simpler due to the inherent reduction of the redundancy among the connection weights. The network topology is then refined using the training data. Scope for growing hidden nodes and pruning links, when necessary (as determined by the network performance), enables the generation of a near optimal network architecture with improved classification performance.

A. Knowledge Encoding

Let an interval $[F_{j_1}, F_{j_2}]$ denote the range of feature F_j covered by class C_k. Then we denote the membership value of the interval as $\mu([F_{j_1}, F_{j_2}]) = \mu$ (between F_{j_1} and F_{j_2}) and compute it as [13]

$$\begin{aligned}&\mu(\text{between } F_{j_1} \text{ and } F_{j_2}) \\ &\quad = \{\mu(\text{greater than } F_{j_1}) * \mu(\text{less than } F_{j_2})\}^{1/2}\end{aligned} \tag{4}$$

where

$$\begin{aligned}\mu(\text{greater than } F_{j_1}) &= \{\mu(F_{j_1})\}^{1/2} \quad \text{if } F_{j_1} \le c_{\text{prop}} \\ &= \{\mu(F_{j_1})\}^2 \quad \text{otherwise}\end{aligned} \tag{5}$$

and

$$\begin{aligned}\mu(\text{less than } F_{j_2}) &= \{\mu(F_{j_2})\}^{1/2} \quad \text{if } F_{j_2} \ge c_{\text{prop}} \\ &= \{\mu(F_{j_2})\}^2 \quad \text{otherwise.}\end{aligned} \tag{6}$$

Here c_{prop} denotes c_{j_l}, c_{j_m}, and c_{j_h} for each of the corresponding three overlapping fuzzy sets *low*, *medium,* and *high* as in [10]. The output membership for the corresponding class C_k is found using (3). Note that, for the computation of z_{ik} [14] of (3), F_{ij} is replaced by the mean of the interval $[F_{j_1}, F_{j_2}]$ of the jth feature.

We have also considered the intervals in which a class is *not* included. The complement of the interval $[F_{j_1}, F_{j_2}]$ of the feature F_j is the region where the class C_k does not lie and is defined as $[F_{j_1}, F_{j_2}]^c$ (where S^c denotes the complement of S). The linguistic membership values for $[F_{j_1}, F_{j_2}]^c$ is denoted by $\mu([F_{j_1}, F_{j_2}]^c) = \mu$ (not between F_{j_1} and F_{j_2}) and is calculated as

$$\begin{aligned}&\mu(\text{not between } F_{j_1} \text{ and } F_{j_2}) \\ &\quad = \max\{\mu(\text{less than } F_{j_1}), \mu(\text{greater than } F_{j_2})\}\end{aligned} \tag{7}$$

since not between F_{j_1} and $F_{j_2} \equiv$ less than F_{j_1} OR greater than F_{j_2}.

Let the linguistic membership values for class C_k in the interval $[F_{j_1}, F_{j_2}]$, as calculated by (4)–(6), be $\{\mu_L([F_{j_1}, F_{j_2}]), \mu_M([F_{j_1}, F_{j_2}]), \mu_H([F_{j_1}, F_{j_2}])\}$. Similarly for the complement of the interval, using (7), we have

$$\{\mu_L([F_{j_1}, F_{j_2}]^c), \mu_M([F_{j_1}, F_{j_2}]^c), \mu_H([F_{j_1}, F_{j_2}]^c)\}.$$

A fuzzy MLP with only one hidden layer is considered, taking two hidden nodes corresponding to $[F_{j_1}, F_{j_2}]$ and its

complement, respectively. Links are introduced between the input nodes and the corresponding nodes in the hidden layer

$$\text{iff } f\mu_A([F_{j_1}, F_{j_2}]) \text{ or } \mu_A([F_{j_1}, F_{j_2}]^c) \geq 0.5 \forall j$$

where $A \in \{L, M, H\}$. The weight $w^{(0)}_{k_{\alpha_p} j_m}$ between the k_{α_p} node of the hidden layer (the hidden node corresponding to the interval $[F_{j_1}, F_{j_2}]$ for class C_k) and $j_m (m \in \{\text{first}(L), \text{second}(M), \text{third}(H)\})$th node of the input layer corresponding to feature F_j is set by

$$w^{(0)}_{k_{\alpha_p} j_m} = p_k + \epsilon \tag{8}$$

where p_k is the *a priori* probability of class C_k and ϵ is a small random number. This hidden node is designated as *positive* node. A second hidden node k_{α_n} is considered for the complement case and is termed a *negative* node. Its connection weights are initialized as

$$w^{(0)}_{k_{\alpha_n} j_m} = (1 - p_k) + \epsilon. \tag{9}$$

Note that the small random number ϵ is considered to destroy any symmetry among the weights. Thus for an l-class problem domain we have 2l nodes in the first hidden layer. In our algorithm we have considered the following two cases.

- *All* connections between these 2l hidden nodes and all nodes in the input layer are possible. The other weights are initially set as small random numbers.
- Only those *selected* connection weights initialized by (4)–(9) are allowed.

It is to be mentioned that the method described above can suitably handle convex pattern classes only. In case of concave classes we consider multiple intervals for a feature F_j corresponding to the various convex partitions that may be generated to approximate the given concave decision region. This also holds for the complement of the region in F_j in which a particular class C_k is not included. Hence, in such cases we introduce hidden nodes, *positive* and *negative*, for each of the intervals with connections being established by (8) and (9) for the cases of a class belonging and not belonging to a region, respectively, such that we get multiple hidden nodes for each of the two cases. In this connection it is to be noted that a concave class may also be subdivided into several convex regions as in [15].

Let there be $(k_{\text{pos}} + k_{\text{neg}})$ hidden nodes, where $k_{\text{pos}} = \Sigma_{\alpha_p} k_{\alpha_p}$ and $k_{\text{neg}} = \Sigma_{\alpha_n} k_{\alpha_n}$, generated for class C_k such that $k_{\text{pos}} \geq 1$ and $k_{\text{neg}} \geq 1$. Now connections are established between kth output node (for class C_k) and only the corresponding $(k_{\text{pos}} + k_{\text{neg}})$ hidden nodes. We assume that if any feature value (for class C_k) is outside some interval α, the total input received by the corresponding hidden node k_α is zero and this thereby produces an output $y^{(1)}_{k_\alpha} = 0.5$ due to the sigmoid nonlinearity of (1).

The connection weight $w^{(1)}_{kk_\alpha}$ between the kth output node and the k_αth hidden node is calculated from a series of equations generated as below. For an interval α as input for class C_k, the expression for output $y^{(2)}_k$ of the kth output node

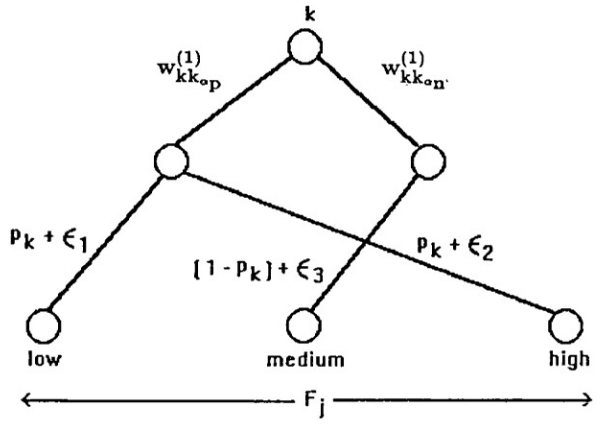

Fig. 1. An example to demonstrate knowledge encoding.

is given by

$$y^{(2)}_k = f(y^{(1)}_{k_\alpha} w^{(1)}_{kk_\alpha} + \sum_{r \neq \alpha} 0.5 w^{(1)}_{kk_r}) \tag{10}$$

where $f(\cdot)$ is the sigmoid function as in (1) and the hidden nodes k_r correspond to the intervals not represented by the convex partition α. Thus for a particular class C_k we have as many equations as the number of intervals (including *not*) used for approximating any concave and/or convex decision region C_k. Thereby, we can uniquely compute each of the connection weights $w^{(1)}_{kk_\alpha}\ \forall \alpha$ (corresponding to each hidden node k_α and class C_k pair).

The network architecture, so encoded, is then refined by training it on the pattern set supplied as input. In case of "all connections" between input and hidden layers, all the link weights are trained. In case of "selected connections" only the selected link weights are trained, while the other connections are kept clamped at zero. If the network achieves satisfactory performance, the classifier design is complete. Otherwise, we resort to node growing or link pruning.

B. An Example

Consider the network depicted in Fig. 1. Let the output node k corresponding to a class C_k be connected to two hidden nodes k_{α_p} and k_{α_n} via connection weights $w^{(1)}_{kk_{\alpha_p}}$ and $w^{(1)}_{kk_{\alpha_n}}$. Let class C_k lie in the interval $[F_{j_1}, F_{j_2}]$ of input feature F_j. Then the weights between the input and the hidden layer are initially set from (8) and (9) as

$$w^{(0)}_{k_{\alpha_p} j_L} = p_k + \epsilon_1$$
$$w^{(0)}_{k_{\alpha_p} j_H} = p_k + \epsilon_2$$

and

$$w^{(0)}_{k_{\alpha_n} j_M} = (1 - p_k) + \epsilon_3.$$

In the case of the network with *all connections* the other weights between the input and the hidden layers (e.g., $w^{(0)}_{k_{\alpha_p} j_M}, w^{(0)}_{k_{\alpha_n} j_L}, w^{(0)}_{k_{\alpha_n} j_H}$) are initialized by small random values. On the other hand, in the case of the network with *selected connections*, these are not considered at all.

Substituting (10) in (1), we have

$$y_k^{(2)} = \frac{1}{1+\exp\{-(y_{k_{\alpha_p}}^{(1)} w_{kk_{\alpha_p}}^{(1)} + 0.5 w_{kk_{\alpha_n}}^{(1)})\}} \tag{11}$$

where $y_k^{(2)}$ is the output of the kth output node and $y_{k_{\alpha_p}}^{(1)}$ is that of the hidden node, corresponding to the presented interval, connected to the kth output node. From (11) we have

$$y_{k_{\alpha_p}}^{(1)} w_{kk_{\alpha_p}}^{(1)} + 0.5 w_{kk_{\alpha_n}}^{(1)} = \ln \frac{y_k^{(2)}}{1-y_k^{(2)}}. \tag{12}$$

Similarly, considering the complement-interval $[F_{j_1}, F_{j_2}]^c$ of the feature F_j, we can write

$$1 - y_k^{(2)} = \frac{1}{1+\exp\{-(0.5 w_{kk_{\alpha_p}}^{(1)} + y_{k_{\alpha_n}}^{(1)} w_{kk_{\alpha_n}}^{(1)}\}}. \tag{13}$$

Therefore

$$0.5 w_{kk_{\alpha_p}}^{(1)} + y_{k_{\alpha_n}}^{(1)} w_{kk_{\alpha_n}}^{(1)} = \ln \frac{1-y_k^{(2)}}{y_k^{(2)}}. \tag{14}$$

The outputs $y_{k_{\alpha_p}}^{(1)}$ and $y_{k_{\alpha_n}}^{(1)}$ are calculated using (1) with appropriate input values. Then from (12) and (14) we can evaluate $w_{kk_{\alpha_p}}^{(1)}$ and $w_{kk_{\alpha_n}}^{(1)}$.

C. Pruning

A large number of connection weights in a network often results in redundancy, leading to the problem of just memorizing the patterns. In such cases pruning of less important links and/or hidden nodes is incorporated in order to get a near optimal network architecture and thereby enhance the generalization capability. There exists various algorithms for pruning [16], [17] ANN's. Here we have incorporated link pruning of the knowledge-based network in a slightly different way.

A connection weight is pruned if its contribution toward the network output is least significant during the presentation of the training set. Therefore, the link $w_{ji}^{(h)}$ in layer (h) is pruned if

$$\sum_p w_{ji}^{(h)} y_i^{(h)} = \min_{k,m} \left\{ \sum_p w_{km}^{(h)} y_m^{(h)} \right\} \tag{15}$$

where the summation is taken over all the patterns p in the training set and the *minimum* is computed over the indices k, m.

When a network with large number of connection weights results in poor classification performance after a certain number of epochs, links between layers $(h+1)$ and (h), for each (h), need to be selected for pruning by (15). The resulting network is retrained for a few more epochs and this process is continued till we get a satisfactory recognition score.

Note that, we do not resort to node pruning as the number of hidden nodes are initially encoded with the domain knowledge and are, therefore, not redundant. During refinement by training, it is the growth of extra links that leads to redundancy. Hence it is our objective to prune a few such redundant links to improve the generalization capability of the network.

D. Growing of Hidden Nodes

If after a certain number of epochs (experimentally determined) the classifier still does not recognize a certain class C_k well and the network size is not too large, we resort to adding a hidden node (instead of pruning) to our knowledge-based model. Connection weights are established between this new node and all the classes. Links are also introduced from all input nodes to this newly added node. Now training is allowed on these new connection weights for a few epochs (again empirically set) using only those samples which are in class C_k while keeping all the other links frozen. Then all the links are retrained with the entire training set and the process of adding, freezing, and retraining are continued, until all the classes are reasonably well recognized. Note that, other approaches for growing of nodes in ANN's may be found in [18] and [19].

IV. Rule Generation

The trained knowledge-based network is used for rule generation in *if–then* form in order to justify any decision reached. These rules describe the extent to which a test pattern belongs or does not belong to one of the classes in terms of antecedent and consequent clauses provided in natural form. We use two rule-generation strategies as described below. The algorithms are, however, different from that reported in [11].

Method (i) Treating the network as a black-box and using the training set input (in numeric and/or linguistic forms) and network output (with confidence factor) to generate the antecedent and consequent parts.

Method (ii) Backtracking along maximal weighted paths using the trained net and utilizing its input and output activations (with confidence factor) for obtaining the antecedent and consequent clauses.

A. Using Numeric and/or Linguistic Inputs—Method (i)

In this method we use an exhaustive set of numeric and/or linguistic inputs along with their hedges at the input for antecedent clauses (*if* parts). We have a total of 9^n patterns (corresponding to *very, mol,* and *not* for each of linguistic values *low, medium,* and *high* of each of the features) for a data set with n features. These patterns constitute the antecedent part of the rules. In the case of numeric patterns, the distance between the pth pattern and each of the linguistic pattern vectors are calculated. The linguistic pattern closest to the pth pattern determines the antecedent part of the rule [11].

To generate the consequent part of the rule, we use a measure which reflects the amount of difficulty in arriving at a decision by minimizing the ambiguity in the computed output vector. A *confidence factor* (CF) is defined [13] as

$$\text{CF} = \frac{1}{2}\left[\{y_{\max}^{(2)}\}^{f_{\max}} + \frac{1}{l-1} \sum_{j=1}^{l} \{y_{\max}^{(2)} - y_j^{(2)}\} \right]$$
$$0 \le \text{CF} \le 1 \tag{16}$$

where $y^{(2)}_{\max} = \max^{l}_{j=1} \{y^{(2)}_j\}, y^{(2)}_j$ is the jth component in the output vector $\boldsymbol{y}^{(2)}$ [by (1)], and $f_{\max}$ indicates the number of occurrences of $y^{(2)}_{\max}$ in $\boldsymbol{y}^{(2)}$. Note that CF takes care of the fact that the difficulty in assigning a particular pattern class depends not only on the highest entry in the output vector $y^{(2)}_{\max}$ but also on its differences from the other entries $y^{(2)}_j$. It is seen that the higher the value of CF, the lower is the difficulty in deciding a class and hence greater is the degree of certainty of the output decision. Based on the value of CF, the system makes the following decisions while generating the consequent clause (*then* part) of the rule. Let $y^{(2)}_k = y^{(2)}_{\max}$ such that the pattern under consideration belongs to class C_k. We have

1) if $(0.8 \leq \mathrm{CF}_k \leq 1.0)$ then very likely class C_k, and there is no second choice;
2) if $(0.6 \leq \mathrm{CF}_k < 0.8)$ then likely class C_k, and there is second choice;
3) if $(0.4 \leq \mathrm{CF}_k < 0.6)$ then mol likely class C_k, and there is second choice;
4) if $(0.1 \leq \mathrm{CF}_k < 0.4)$ then not unlikely class C_k, and there is no second choice;
5) if $(\mathrm{CF}_k < 0.1)$ then unable to recognize class C_k, and there is no second choice.

To obtain a second choice corresponding to a pattern class C_{k_2} (say), we find the *confidence factor* CF_{k_2} for the second highest entry $y^{(2)}_{k_2}$ in the output vector using (16). There may be some cases where there are multiple entries with the highest value $y^{(2)}_{\max}$ in the output vector. In that case, there will not be a second choice of pattern class. Instead, the form of the consequent will be "*likely* class C_k *or* C_j" where the output values corresponding to classes C_k and C_j both have the highest value $y^{(2)}_{\max}$.

Identical rules, if any, are discarded from the generated rule set.

B. Backtracking Along Trained Connection Weights—Method (ii)

An input pattern $\boldsymbol{F}_p$ from the training set is presented to the input of the trained network and its output computed. The consequent part of the corresponding *if–then* rule is generated by (16) as described in Section IV-A. To find the antecedent clauses of the rule, we backtrack from the output layer to the input through the maximal weighted links. The path from node k in the output layer to node i_A in the input layer through node j in the hidden layer is maximal if

$$w^{(1)}_{kj} y^{(1)}_j + w^{(0)}_{ji_A} y^{(0)}_{i_A} = \max_m \{w^{(1)}_{km} y^{(1)}_m + w^{(0)}_{mi_A} y^{(0)}_{i_A}\} \quad (17)$$

provided that $y^{(1)}_j \geq 0.5, y^{(0)}_{i_A} > 0.5$, and the *maximum* is computed over the index m. Here the path length from node k in the output layer to node j in the hidden layer is $w^{(1)}_{kj} y^{(1)}_j$ and not $w^{(1)}_{kj}$ as defined by Mitra and Pal [11] in an earlier approach. Besides, the CF of (16) is also different and in certain ways better than the *belief* used there. We consider only one node i_A corresponding to the three linguistic values of each feature F_i so that

$$w^{(0)}_{ji_A} y^{(0)}_{i_A} = \max_{B \in \{L,M,H\}} w^{(0)}_{ji_B} y^{(0)}_{i_B} \quad (18)$$

where A and B correspond to low (L), medium (M) or high (H). The 3-D linguistic pattern vector *low, medium* or *high* with or without hedges [corresponding to the linguistic feature F_{i_A} computed by (18)], which is closest to the relevant 3-D part of pattern $\boldsymbol{F}_p$, is selected as the antecedent clause [11]. This is done for all input features to which a path may be found by (17). The complete *if* part of the rule is found by ANDing the clauses corresponding to each of the features, e.g.,

If F_1 is mol A and F_2 is not A and
$\cdots$ and F_n is very A.

1) Negative Rules: It may sometimes happen that we are unable to classify a test pattern directly with the help of the *positive* rules (concerning its belongingness to a class) derived by any of the above two methods. In such cases, we proceed by discarding some classes which are unlikely to contain the pattern, and thereby arrive at the class(es) to which the pattern possibly belongs. In other words, in the absence of positive information regarding the belongingness of pattern $\boldsymbol{F}_p$ to class C_k, we use the complementary information about the pattern $\boldsymbol{F}_p$ not belonging to class $C_{k'}$. To handle such situations, we have generated *negative* rules with the consequent part of the form *not in class* $C_{k'}$ by backtracking from the output layer through the trained connection weights. Note that, for *positive* rules we traverse the hidden node k_{α_p} while for *negative* rules we backtrack along the hidden node k_{α_n}.

Let an input pattern $\boldsymbol{F}_p$ from the training set be presented to the input layer of the trained network such that the output of the node in the output layer corresponding to the class $C_{k'}$ is minimum, i.e., $y^{(2)}_{k'} = \min_l \{y^{(2)}_l\}$. Therefore, we are certain that the pattern is (possibly) not included in the class $C_{k'}$. Hence, the consequent part of the corresponding rule becomes *not in class* $C_{k'}$. The antecedent part of the rule is obtained by backtracking from the output node k' through the maximal path using (17) with the restrictions that now we consider the absolute values of the individual product terms. The corresponding rule, so obtained, is of the form

If F_1 is mol A and $\cdots$ and F_n is very A
then the pattern is not in class $C_{k'}$.

Note that the approach in [11] did not consider such negative rules. ♣

It is worth mentioning that the above rule generation techniques can also handle the situations where the input is given in set form. In other words, the feature information F_j of a test pattern is neither linguistic nor numeric, but may be available as 1) $F_j \geq F_{j_1}$, some lower bound; 2) $F_j \leq F_{j_2}$, some upper bound; or 3) in some interval $[F_{j_1}, F_{j_2}]$ such that F_j lies between F_{j_1} and F_{j_2}. In these cases, the linguistic values *low, medium, high* corresponding to the input given in the set form and hedges are evaluated from (4)–(6). Then the rule with this antecedent is picked up, and we take a decision on the basis of the corresponding consequent part regarding its class (as explained earlier).

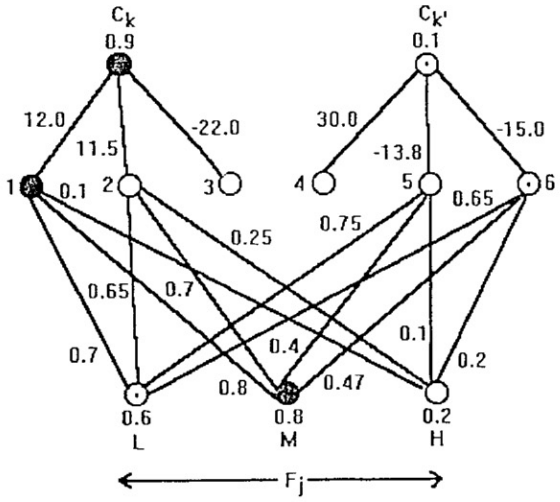

Fig. 2. An example to demonstrate *positive* and *negative* rule generation using method (ii).

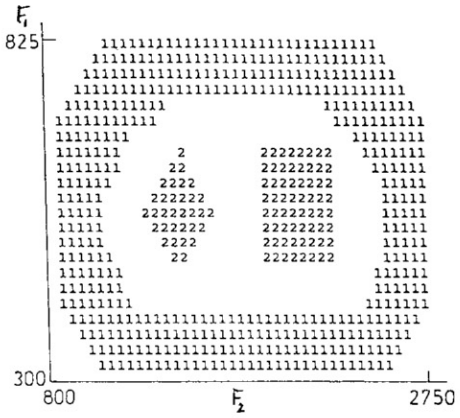

Fig. 3. Pattern set **Pat1**.

2) An example: Consider a knowledge-based network (Model AN) given in Fig. 2 demonstrating the rule generation technique using method (ii). We have considered the feature F_j for the antecedent clause. Only the classes C_k and $C_{k'}$ with maximum and minimum outputs, respectively (on presenting a pattern at the input layer), have been considered for the generation of *positive* and *negative* rules. The hidden nodes 1 and 2 correspond to two convex segments of the region represented by the class C_k, and 3 corresponds to the region complement to class C_k. Similarly, for the class $C_{k'}$, we have considered the hidden node 4 corresponding to the region of the class $C_{k'}$, and 5 and 6 correspond to the region other than class $C_{k'}$. Thus, nodes 1 and 2, and 4 represent the *positive* nodes for classes C_k and $C_{k'}$, respectively. Similarly, the nodes 3, 5, and 6 denote the *negative* nodes for classes C_k and $C_{k'}$, respectively.

A pattern with linguistic values $\text{low}(L) = 0.6, \text{medium}(M) = 0.8$, and $\text{high}(H) = 0.2$ of feature F_j is presented to the input layer of the network. Assume that the activations of the output nodes corresponding to the classes C_k and $C_{k'}$ are 0.9 and 0.1, respectively. Therefore, backtracking starts from the output node corresponding to the class C_k and searches for the maximal path through the hidden nodes 1 and 2 only (if the activations of these nodes are at least 0.5) for *positive* rule generation. For the *negative* rule generation, it starts from the output node corresponding to the class $C_{k'}$ and searches for the maximal path through the hidden nodes 5 and 6 only (if the outputs of these nodes are at least 0.5). The links with weights as shown in the figure are obtained during training. For clarity of the figure, we have not considered the links from hidden node 3 to L, M, and H corresponding to the feature F_j as we do not require these links for the generation of *positive* rule. Similarly, the links from the hidden node 4 to the input nodes L, M, and H are not shown.

We denote the path length [as explained by (17)] from the hidden node j to the input node i by $\text{path}_{ji}^{(0)}$, and that from the output node k to the hidden node j by $\text{path}_{kj}^{(1)}$. The path length from the output node k to the input node i via hidden node j is denoted by path_{kji}. Therefore, from Fig. 2 we find the following path lengths: $\text{path}_{1L}^{(0)} = 0.7 * 0.6 = 0.42, \text{path}_{1M}^{(0)} = 0.8 * 0.8 = 0.64, \text{path}_{2L}^{(0)} = 0.39$ and $\text{path}_{2M}^{(0)} = 0.56$. Note that, $\text{path}_{1H}^{(0)}$ and $\text{path}_{2H}^{(0)}$ have not been considered as the activation of the input node H is less than 0.5. The total inputs received by the hidden nodes 1 and 2 are found to be 1.08 and 1.0, respectively. Therefore, the activations of these nodes are $y_1^{(1)} = 0.75$ and $y_2^{(1)} = 0.73$, respectively by (1). The path lengths between the nodes in the output layer and the hidden layer are $\text{path}_{k1}^{(1)} = 9.0$ and $\text{path}_{k2}^{(1)} = 8.4$. The total path lengths are found to be $\text{path}_{k1L} = 9.42, \text{path}_{k1M} = 9.64, \text{path}_{k2L} = 8.79, \text{path}_{k2M} = 8.96$. Hence, the maximal path from the output node corresponding to the class C_k is obtained as the path via the hidden node 1 to the input node M (the selected path consists of the links joining the nodes indicated by solid circles in Fig. 2). The antecedent clause corresponding to the feature F_j is "F_j *is mol medium*," and is obtained by finding the closest match of the 3-D vector corresponding to the feature F_j, to the respective linguistic pattern (with/without hedges). The consequent part of the rule is *very likely class* C_k, as obtained from (16).

For the generation of *negative* rule, we compute the following path lengths: $\text{path}_{5L}^{(0)} = 0.45, \text{path}_{5M}^{(0)} = 0.32, \text{path}_{6L}^{(0)} = 0.39, \text{path}_{6M}^{(0)} = 0.38$. The total inputs received by the hidden nodes 5 and 6 are 0.79 and 0.81, respectively. Therefore, the corresponding activations are $y_5^{(1)} = 0.69$ and $y_6^{(1)} = 0.69$. Thus, the path lengths between the nodes in the output layer and the hidden layer are $\text{path}_{k'5}^{(1)} = 9.52$ and $\text{path}_{k'6}^{(1)} = 10.35$. The total path lengths are $\text{path}_{k'5L} = 9.97, \text{path}_{k'5M} = 9.84, \text{path}_{k'6L} = 10.74, \text{path}_{k'6M} = 10.73$. Hence, the maximal path from the output node corresponding to the class $C_{k'}$ is obtained as the path via the hidden node 6 to the input node L (the path consisting of the links joining the nodes indicated by circles with dots inside in Fig. 2). Therefore, the corresponding antecedent clause, "F_j *is mol low*," of the *negative* rule is obtained as in the earlier case. Thus, the *negative* rule is: *If* F_j *is mol low then the pattern is not in class* $C_{k'}$.

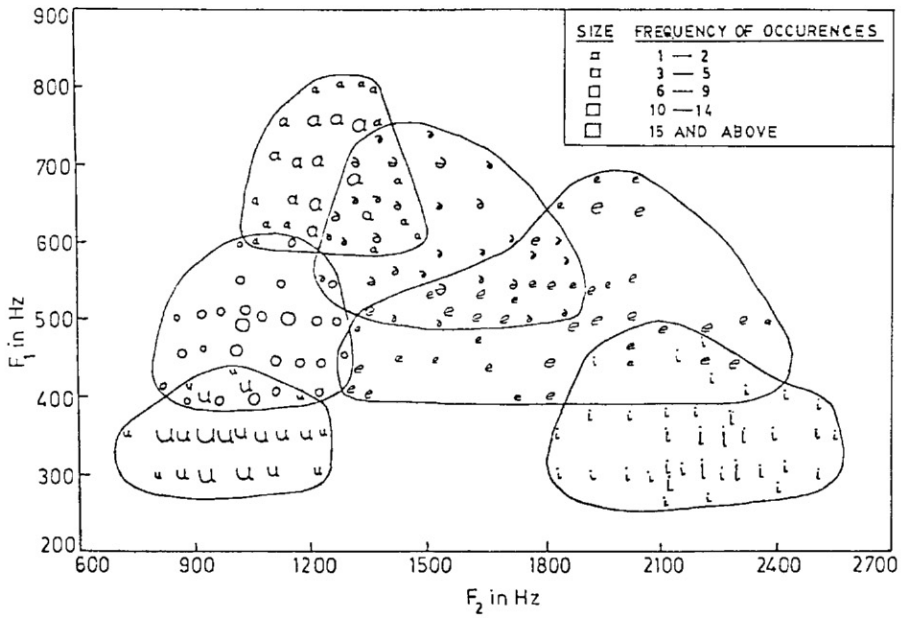

Fig. 4. **Vowel** data.

V. Experimental Results

In this section we compare the classification and rule generation performance of the proposed knowledge-based model with that of the conventional and fuzzy versions of the MLP [10], [11], and the fuzzy min–max neural network [12] both on synthetic and real-life (speech and medical) data. In all the cases the data sets have been divided into two subsets—*training* and *testing*. The synthetic data **Pat1** (Fig. 3) contains two input features, two pattern classes, and consists of 557 pattern points. As the class structures are concave, we have found approximately (by inspecting the feature space) the set of intervals of the features where each of the pattern classes lie (or do not lie). The networks are trained with 10% of the original data while the remaining 90% data constitutes the test set.

The speech data **Vowel** 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 to 35 years. The data consists of 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 plot in the $F_1 - F_2$ plane for ease of depiction. The data contains six vowel classes—∂, a, i, u, e, o represented as 1, 2, 3, 4, 5, and 6 in the sequel. The *training* set contains 10% of the original data set.

The medical data **Hepato**, consisting of nine input features and four pattern classes, deals with various *Hepatobiliary disorders* [20] of 536 patient cases. The input features are the results of different biochemical tests, *viz.*, glutamic oxalacetic transaminate (GOT, Karmen unit), glutamic pyruvic transaminase (GPT, Karmen unit), lactate dehydrase (LDH, iu/l), gamma glutamyl transpeptidase (GGT, mu/ml), blood urea nitrogen (BUN, mg/dl), mean corpuscular volume of red blood cell (MCV, fl), mean corpuscular hemoglobin (MCH, pg), total bilirubin (TBil, mg/dl) and creatinine (CRTNN, mg/dl). The hepatobiliary disorders alcoholic liver damage (ALD), primary hepatoma (PH), liver cirrhosis (LC) and cholelithiasis (C), constitute the four output classes. In this case, 30% of the original data set comprises the *training* set while the remaining 70% data forms the *test* set. In the case of the medical data we have assumed the pattern classes to be convex as it is otherwise very difficult to visualize the exact nature of the nine-dimensional feature space.

It is found that the knowledge-based model converges to a good solution with a very small number of training epochs (iterations) in all the four cases. Note that, we have used the following four knowledge-based models designated as follows:

all connections with not:	Model AN
all connections without not:	Model A
selected connections with not:	Model SN
selected connections without not:	Model S.

Results are compared with those of the fuzzy MLP (Model F), the conventional MLP (Model C), and the fuzzy min–max network (Model FMM) [12]. The number of links required in each case is appropriately indicated (in parenthesis after the name of the corresponding model) in the tables. The variables f_d and f_e of (3) were set at 5.0 and 1.0, respectively [10], for the speech and medical data. For the synthetic data we use $d_k \in \{0, 1\}$ and hence f_d and f_e are not required.

A. Classification

Table I depicts the result obtained with **Pat1** data. A total of six intervals (i.e., six hidden nodes) for the two features are found to be sufficient to characterize the classes if we do not consider the intervals in which any of the classes is *not included* [(4)–(6) only]. This is termed as the *without not* case. Otherwise, if we consider both the intervals of the features in which a class *is included* and *is not included*, we require a

TABLE I
PERFORMANCE OF DIFFERENT MODELS ON **Pat1** DATA

Model	Class	Score(%)	
		Training	Testing
AN (82)	1	100.0	100.0
	2	0.0	0.0
	Overall	83.64	82.47
A (47)	1	100.0	100.0
	2	100.0	100.0
	Overall	100.0	100.0
SN ($\leq$ 82)	1	100.0	100.0
	2	100.0	100.0
	Overall	100.0	100.0
S ($\leq$ 47)	1	100.0	100.0
	2	0.0	0.0
	Overall	83.64	82.47
FMM (84)	1	100.0	100.0
	2	100.0	84.09
	Overall	100.0	97.21
FMM (48)	1	100.0	100.0
	2	55.56	48.86
	Overall	92.73	91.04

TABLE II
PERFORMANCE OF DIFFERENT MODELS ON **Vowel** DATA

Model	Class	Score(%)	
		Training	Testing
AN (138)	1	42.86	27.69
	2	87.5	86.42
	3	94.12	87.74
	4	100.0	82.35
	5	90.0	69.52
	6	100.0	93.83
	Overall	90.59	78.63
SN ($\leq$ 138)	1	0.0	0.0
	2	62.5	58.02
	3	94.12	87.74
	4	100.0	82.35
	5	85.0	68.45
	6	94.44	93.21
	Overall	82.35	73.79
FMM (504)	1	71.43	30.77
	2	100.0	80.25
	3	100.0	91.61
	4	86.67	72.06
	5	95.0	62.57
	6	88.89	85.80
	Overall	91.76	73.92
FMM (198)	1	71.43	75.38
	2	50.0	41.98
	3	76.47	74.19
	4	33.33	43.38
	5	65.0	68.98
	6	44.44	35.19
	Overall	56.47	56.36

total of ten intervals (i.e., ten hidden nodes). This is called the *with not* case. It is observed that models A and SN give 100% recognition score in just 600 epochs. The other models (AN and S) have not been able to recognize class 2 at this stage. In model AN, perhaps the large number of interconnections encode too much redundant information thereby not enabling the classifier to recognize class 2. On the other hand, model S provides poor result probably due to under-information. The performance of C and F is the same as that of AN and S. That is why we have not included the results for C and F in Table I.

We have resorted to pruning of links in models AN, F and growing of hidden nodes in cases of models S, F, C. It is found that after only 100 epochs of growing the model SN provides overall recognition score of 100% on the training set and 99.8% on the test set. This demonstrates a remarkable improvement in performance. Hidden nodes were also added to models C and F at the same stage but the performance is found to be poor (0% recognition for class 2) in case of **Pat1** data. Pruning model AN resulted in 100% recognition scores for both the training and test sets. The links were pruned from 600 epochs at intervals of ten epochs, up to 750 epochs, and then the network was trained until 900 epochs. Although model F could now recognize around 20% patterns from class 2, this was considerably less than that by model AN.

Table II shows the results obtained with **Vowel** data. Since all the classes in the feature space are convex, we use two hidden nodes for each of the classes. Hence only *with not* models have been considered and we require a total of 12 hidden nodes for this data set. The results demonstrate that model AN gives acceptably good performance in just 200 epochs whereas model SN cannot do the same due to under-information. Note that, the vowel classes are overlapping and fuzzy, thereby generating fuzzy output class membership values that require storage of more information than in case of crisp class membership values. Perhaps this accounts for the better performance of model AN (with more connections). Models C and F were unable to recognize classes 1, 2, and 4, and fared the worst (overall recognition score during training and testing being 42.35 and 39.19% for the model C, and 55.29 and 52.93% for the model F). As before, their details are not mentioned in the table to restrict the size of the article.

As model AN performed reasonably well for all classes initially (before growing), the incorporation of additional hidden nodes did not improve the results in this case. However, when model SN was augmented for class 1 it was found that after 350 epochs the model could recognize 14.29% of class 1 during training and 1.54% during testing. The overall scores rose to 83.53 and 74.17% for the training and testing sets, respectively. The results are depicted in Table III.

Table IV demonstrates the classification performance for the medical data **Hepato** where classes 1, 2, 3, and 4 correspond to the four disease classes ALD, PH, LC, and C, respectively. Note that here we do not know *a priori* the shape of the pattern classes in the nine-dimensional feature space. We have

TABLE III
EFFECT OF ADDING HIDDEN NODE ON THE PERFORMANCE OF THE VARIOUS KNOWLEDGE-BASED MODELS ON **Vowel** DATA

Model	Class	Score(%)	
		Training	Testing
AN	1	42.86	23.08
	2	87.5	88.89
	3	94.12	87.74
	4	100.0	82.35
	5	90.0	70.05
	6	100.0	93.83
	Overall	90.59	78.63
SN	1	14.29	1.54
	2	62.5	58.02
	3	94.12	92.26
	4	100.0	84.56
	5	85.0	66.84
	6	94.44	93.83
	Overall	83.53	74.17

assumed that the classes are convex, so that only eight hidden nodes are used corresponding to the four classes. As in the case of **Vowel**, only the knowledge-based models *AN* and *SN* have been used. Since we have approximated the structures of the classes as convex, model SN which uses only those links that are encoded with the initial knowledge performs rather poorly. Perhaps it would require more nodes and links than were available under our assumption. However, model AN, which is allowed to grow extra links, is found to have solved this problem. Its performance is considerably better than that of models SN and F in just 500 epochs (Table IV). Note that, the model C, being unable to recognize classes 1, 3, and 4 is not included in the table.

Tables I, II, and IV also show the classification performance of the fuzzy min-max neural network (model FMM) [12] on the three data sets. In this model, the number of links can be varied by altering some of the parameters. Here we show the results for two different configurations *viz.*, 1) providing mol the same overall recognition score (on the training sets) as the proposed model and 2) providing mol the same number of links as the proposed model. Note that, the model A for **Pat1** (Table I), and model AN for both **Vowel** (Table II) and **Hepato** (Table IV) have been compared for this purpose, as they perform the best. It is clear that the model FMM requires more links than the proposed model to get mol the same overall recognition score. Similarly, with mol the same number of links the model FMM performs poorer for all the data sets.

Figs. 5 and 6 depict the variation of *mean square error* with the number of sweeps for pattern sets **Pat1** and **Vowel**, respectively. In the case of **Pat1** we demonstrate the behavior of the two better knowledge-based models A and SN only, for ease of explanation. It is observed that model C has the worst performance. Model A (for **Pat1**) and model AN (for **Vowel**) behave the best. It is found that Model SN is better than Model F in the beginning and converges to a good solution very fast (in about 600 sweeps in Fig. 5) for **Pat1** while Model

TABLE IV
PERFORMANCE OF DIFFERENT MODELS ON **Hepato** DATA

Model	Class	Score(%)	
		Training	Testing
F (260)	1	41.18	37.80
	2	90.57	89.60
	3	2.70	9.20
	4	91.43	92.77
	Overall	59.75	60.48
AN (236)	1	61.76	52.44
	2	84.91	77.60
	3	59.46	43.68
	4	91.43	86.75
	Overall	75.47	66.31
SN ($\leq$ 236)	1	0.0	0.0
	2	98.11	100.0
	3	0.0	0.0
	4	0.0	0.0
	Overall	32.70	33.16
FMM (2068)	1	76.47	32.93
	2	90.57	60.0
	3	48.65	2.30
	4	82.86	55.42
	Overall	76.10	39.79
FMM (236)	1	8.82	0.0
	2	100.0	100.0
	3	18.92	0.0
	4	0.0	0.0
	Overall	39.62	33.16

F requires about twice to thrice this time to reach the same level of performance. In contrast, for **Vowel** data, Model F surpasses model SN at around 500 sweeps (as seen from Fig. 6). However, Model AN/A is always the best perhaps due to the presence of less redundancy (than Model F) along with more knowledge (than Model SN). Note that, Tables I and II depict the performance of the knowledge-based models at 600 sweeps (epochs), respectively. This accounts for the relatively poor performance of Model F at this stage, whereas it fares better with longer training time (as is evident from the figures).

B. Rule Generation

Tables V and VI compare the rules generated for **Pat1** and **Vowel** data, respectively, by the methods described in Section IV using the proposed knowledge encoded networks and the fuzzy MLP [10]. The rules generated on the various models are not identical due to the different amounts of redundancy inherent in them and the difference in the encoding of their network architectures. Method (ii) often produces different results (as compared to method (i) where only the input and output of the neural net are considered) because here the trained connection weight magnitudes are utilized during the tracing of the maximal weighted paths, thereby using the encoded and refined domain knowledge along with the test case feature values.

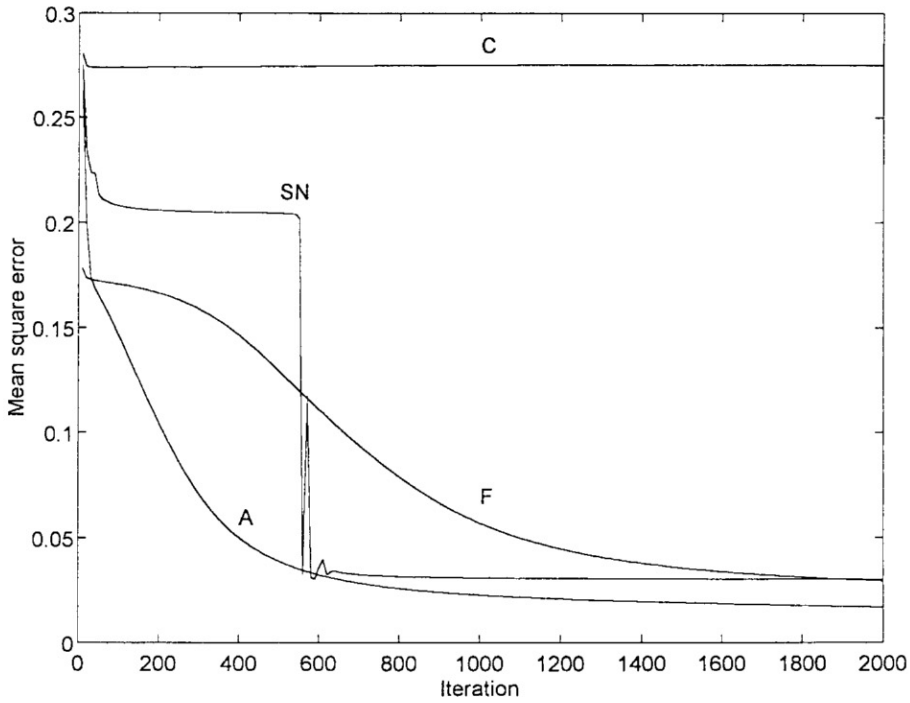

Fig. 5. Variation of *mean square error* with *number of sweeps* for **Pat1**.

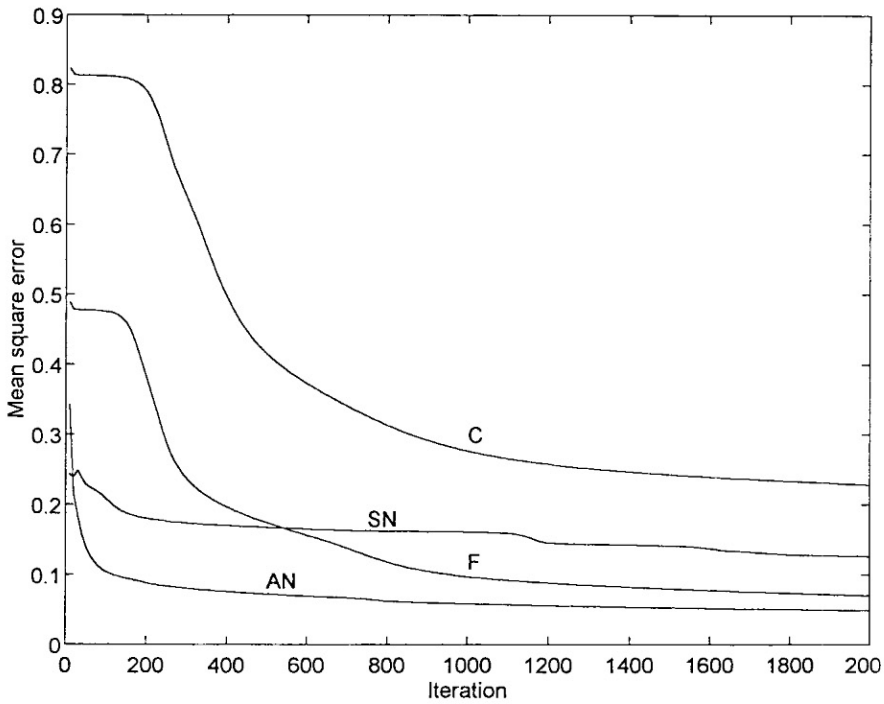

Fig. 6. Variation of *mean square error* with *number of sweeps* for **Vowel**.

1) Negative Rules: Let us consider the trained connection weights $w_{kk_\alpha}^{(1)}$ of the knowledge-based network in the case of **Pat1** to explain the generation of *negative* rules. It is interesting to note that the weights $w_{kk_{\alpha_n}}^{(1)}$ connecting k_{α_n} nodes in the hidden layer with the corresponding kth output node are found to be negative, whereas those connecting k_{α_p} nodes are positive for each of the classes C_k. Therefore, when a pattern belonging to class $C_{k'}$ is presented to the input layer of the network, the output produced by the k_{α_n} hidden nodes is greater than those by k_{α_p} hidden nodes (or sometimes comparable in magnitude when the weights $w_{kk_{\alpha_p}}^{(1)}$ and $w_{kk_{\alpha_n}}^{(1)}$ are also comparable). But in such cases, the output produced by the k'_{α_p} nodes is always found to be greater than those by k'_{α_n} nodes. The hidden nodes 1, 2, 3, and 4 correspond to the intervals to which class 1 belongs, while node 5 refers to the interval in which class

TABLE V
RULES OBTAINED BY DIFFERENT MODELS FOR **Pat1**

Model	Antecedent	Consequent	
		Method (i)	Method (ii)
A	F_1 low and F_2 high	likely class 1 but unable to recognize class 2	very likely class 1
	F_1 mol medium and F_2 mol medium	mol likely class 2 but unable to recognize class 1	likely class 2 but unable to recognize class 1
SN	F_1 low and F_2 high	very likely class 1	very likely class 1
	F_1 mol medium and F_2 mol medium	likely class 2 but unable to recognize class 1	very likely class 2
F	F_1 low and F_2 high	very likely class 1	very likely class 1
	F_1 mol medium and F_2 mol medium	likely class 2 but unable to recognize class 1	likely class 2 but unable to recognize class 1

1 does not lie. The corresponding connection weights are 8.241 502, 13.473 857, 13.286 455, 8.405 158, 16.488 053, and 16.458 573, respectively. Similarly, hidden nodes 6 and 7 correspond to the intervals to which class 2 belongs, while the hidden nodes 8, 9, and 10 are indicative of the region where class 2 is not included. Their connection weights are −43.487 919, −9.280 226, −14.061 259, and −9.116 443, respectively. Considering this, we backtrack along k'_{α_n} nodes while determining a rule about a pattern not belonging to class $C_{k'}$ and generate that path having the maximal value for the magnitude of the product term (as explained in Section IV-B. for *negative* rules).

Two sample *negative* rules obtained by method (ii) using the AN and F models for **Vowel** data are provided below.

Using model AN:

If F_1 is mol low and F_2 is medium and F_3 is very low then the pattern is not in class 4.

If F_1 is very high and F_2 is mol medium and F_3 is very medium then the pattern is not in class 3;

Using model F:

If F_1 is high and F_2 is medium and F_3 is high then the pattern is not in class 4.

If F_1 is very high and F_2 is mol medium and F_3 is very medium then the pattern is not in class 3.

It is seen that *negative* rules offer an useful solution in cases where no suitable *positive* rule can be found. Note that, model F does not have k_{α_p} or k_{α_n} nodes encoded in its structure. We have provided the negative rules in this case by just backtracking along the maximal magnitude paths from the class producing the minimal output. The rules for model F are provided as an extension to the approach of [11] while also enabling us to make a comparative study. Similarly, a sample *negative* rule generated by method (ii) with the *AN* and *F* models for the medical data **Hepato** is provided below.

Using model AN:

If F_1 is low and F_2 is low and F_3 is very medium and F_4 is low and F_5 is low and F_6 is medium and F_7 is mol medium and F_8 is low and F_9 is very medium then the pattern is not in class 1.

Using model F:

If F_1 is low and F_2 is low and F_3 is very medium and F_4 is low and F_5 is mol medium and F_6 is medium and F_7 is mol medium and F_8 is mol low and F_9 is very medium then the pattern is not in class 1.

VI. CONCLUSIONS AND DISCUSSION

A new methodology of knowledge encoding among the connection weights of a fuzzy MLP [10] is described. This enables the network to perform classification and rule generation more efficiently. It involves development of a technique for generating an appropriate architecture of the fuzzy MLP [10] in terms of hidden nodes and links. Node growing and link pruning are used to enhance performance. It is found that the knowledge-based classification leads to better result than those of the conventional and fuzzy versions of the MLP [10], [11], and the fuzzy min-max neural network [12].

During learning an MLP searches for the set of weights that corresponds to some local minima. There may be a large number of such minimum values corresponding to various *good* solutions. The knowledge-based network initially considers these weights so as to be near one such *good* solution. As a result, the searching space gets reduced and learning becomes faster. Note that, unlike the other methods [1], [2], the proposed knowledge encoding technique involves nonbinary weighting mechanism based on the domain knowledge of a data set. The incorporation of fuzziness at various levels also helps the model to efficiently handle uncertain and ambiguous information both at the input and the output.

Conventional and fuzzy versions of the MLP consider empirically determined fixed architecture, whereas the knowledge-based model automatically determines it. The fuzzy min–max network [12] generates hidden nodes from some empirically determined parameter values. It is observed that this network requires larger number of links than the proposed model to generate mol the same recognition score.

Our model is capable of generating both *positive* and *negative* rules in linguistic form to justify any decision reached.

TABLE VI
RULES OBTAINED BY THE KNOWLEDGE-BASED AND FUZZY MLP FOR **Vowel**

Model	Antecedent	Consequent	
		Method (i)	Method (ii)
AN	F_1 medium, F_2 low and F_3 low	very likely class 6	very likely class 4
	F_1 low, F_2 high and F_3 mol medium	very likely class 3	very likely class 3
	F_1 low, F_2 low and F_3 high	not unlikely class 6	mol likely class 6 but unable to recognize class 4
	F_1 high, F_2 medium and F_3 very low	mol likely class 1 but unable to recognize class 2	not unlikely class 1
	F_1 very high, F_2 low and F_3 very medium	mol likely class 2 but unable to recognize class 5	mol likely class 2 but unable to recognize class 5
	F_1 high, F_2 high and F_3 medium	likely class 5 but unable to recognize class 3	very likely class 5
F	F_1 medium, F_2 low and F_3 low	very likely class 6	very likely class 4
	F_1 low, F_2 high and F_3 mol medium	likely class 3 but not unlikely class 5	likely class 3 but unable to recognize class 5
	F_1 low, F_2 low and F_3 high	mol likely class 6 but unable to recognize class 4	mol likely class 6 but unable to recognize class 4
	F_1 high, F_2 medium and F_3 very low	likely class 1 but unable to recognize class 2	likely class 1 but unable to recognize class 2
	F_1 very high, F_2 low and F_3 very medium	mol likely class 2 but unable to recognize class 1	mol likely class 2 but not unlikely class 1
	F_1 high, F_2 high and F_3 medium	likely class 5 but unable to recognize class 3	very likely class 5

These rules are found to be useful for inferencing in ambiguous cases. Note that, the rule generation algorithms described in this article are different from that derived from fuzzy MLP in [11]. A comparative study with that algorithm has also been provided to support this. It is observed that the less redundant knowledge-based model yields better rules much earlier. The concept of negative rules has been introduced to handle situations where a pattern does not belong to a specific class with high certainty. In such ambiguous situations, the complementary case of a pattern certainly not belonging to a class is considered to provide an appropriate explanation.

REFERENCES

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

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

[3] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, 1987.

[4] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1994.

[5] S. I. Gallant, "Connectionist expert systems," *Commun. Assoc. Computing Machinery*, vol. 31, pp. 152–169, 1988.

[6] ———, *Neural-Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1994.

[7] S. Mitra and S. K. Pal, "Neuro-fuzzy expert systems: Overview with a case study," in *Fuzzy Reasoning in Information, Decision, and Control Systems*, S. G. Tzafestas and A. N. Venetsanopoulos, Eds. Boston, MA: Kluwer, 1994, pp. 121–143.

[8] H. F. Yin and P. Liang, "A connectionist incremental expert system combining production systems and associative memory," *Int. J. Pattern Recognition Artificial Intell.*, vol. 5, pp. 523–544, 1991.

[9] R. J. Machado and A. F. Rocha, "A hybrid architecture for connectionist expert systems," in *Intell. Hybrid Syst.*, A. Kandel and G. Langholz, Eds. Boca Raton, FL: CRC, 1992.

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

[11] S. Mitra and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, 1995.
[12] P. K. Simpson, "Fuzzy min-max neural networks—Part 1: Classification," *IEEE Trans. Neural Network*, vol. 3, pp. 776–786, 1992.
[13] S. K. Pal and D. P. Mandal, "Linguistic recognition system based on approximate reasoning," *Inform. Sci.*, vol. 61, pp. 135–161, 1992.
[14] S. K. Pal and D. D. Majumder, *Fuzzy Mathematical Approach to Pattern Recognition*. New York: Wiley, 1986.
[15] D. P. Mandal, C. A. Murthy, and S. K. Pal, "Determining the shape of a pattern class from sampled points in $\mathcal{R}^2$," *Int. J. General Syst.*, vol. 20, pp. 307–339, 1992.
[16] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, 1993.
[17] J. Sietsma and R. J. F. Dow, "Creating artificial neural network that generalize," *Neural Networks*, vol. 4, pp. 67–79, 1991.
[18] S. G. Romaniuk and L. O. Hall, "Divide and conquer neural networks," *Neural Networks*, vol. 6, pp. 1105–1116, 1993.
[19] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524–532.
[20] Y. Hayashi, "A neural expert system with automated extraction of fuzzy if–then rules and its application to medical diagnosis," in *Advances in Neural Information Processing Systems*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 578–584.

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, India, in 1984, 1987, and 1989, respectively. She received the Ph.D. degree in computer science from the Indian Statistical Institute, Calcutta, in 1995.

She was a Senior Research Fellow of the Council for Scientific and Industrial Research from 1989 to 1991. Since 1991, she has been with the Indian Statistical Institute, as a Programmer. In the period 1992 to 1994 she was with the European Laboratory for Intelligent Techniques Engineering, Aachen, Germany, as a German Academic Exchange Service (DADD) Fellowship holder. Her research interests include pattern recognition, fuzzy sets, artificial intelligence, and neural networks.

From 1978 to 1983, she was a recipient of the National Talent Search Scholarship from the National Council for Educational Research and Training, India. She was awarded the 1994 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award.

Rajat K. De received the B. Tech. degree in computer science and engineering from the University of Calcutta, India, in 1991, and the Master's degree in computer science and engineering from Jadavpur University, India, in 1993. Since October 1993 he has been working as a Research Scholar at the Indian Statistical Institute, Calcutta, India, toward the Ph.D. degree, with a Dr. K. S. Krishnan Senior Research Fellowship, sponsored by the Department of Atomic Energy of India. His research interest includes pattern recognition, fuzzy sets, and neural networks.

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

He is a Professor and Founding Head of Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. 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. His research interests 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 two books *Fuzzy Models for Pattern Recognition* (New York: IEEE Press, 1992) and *Genetic Algorithms for Pattern Recognition* (Boca Raton, FL: CRC Press, 1996).

Dr. Pal received the 1990 Shanti Swarup Bhatnagar Prize in Engineering Sciences, the 1993 Jawaharlal Nehru Fellowship, the 1993 Vikram Sarabhai Research Award, the 1993 NASA Tech Brief Award, the 1994 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award and the 1995 NASA Patent Application Award. He is a Fellow of the IIndian National Science Academy, Indian Academy of Sciences, National Academy of Sciences, India, Indian National Academy of Engineering, Institute of Engineers, India, and the IETE, a Member of the Executive Advisory Editorial Board for the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the *International Journal of Approximate Reasoning*, and an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, *Pattern Recognition Letters*, *Neurocomputing*, *Applied Intelligence*, *Information Sciences: Applications*, the *International Journal of Knowledge-Based Intelligent Engineering Systems*, and the *Far-East Journal of Mathematical Sciences*.