Rough-Fuzzy MLP: Modular Evolution, Rule Generation, and Evaluation

Sankar K. Pal, *Fellow*, *IEEE*, Sushmita Mitra, *Senior Member*, *IEEE*, and Pabitra Mitra, *Student Member*, *IEEE*

Abstract—A methodology is described for evolving a Rough-fuzzy multi layer perceptron with modular concept using a genetic algorithm to obtain a structured network suitable for both classification and rule extraction. The modular concept, based on "divide and conquer" strategy, provides accelerated training and a compact network suitable for generating a minimum number of rules with high certainty values. The concept of variable mutation operator is introduced for preserving the localized structure of the constituting knowledge-based subnetworks, while they are integrated and evolved. Rough set dependency rules are generated directly from the real valued attribute table containing fuzzy membership values. Two new indices viz., "certainty" and "confusion" in a decision are defined for evaluating quantitatively the quality of rules. The effectiveness of the model and the rule extraction algorithm is extensively demonstrated through experiments alongwith comparisons.

Index Terms—Soft computing, knowledge-based fuzzy networks, rough sets, genetic algorithms, pattern recognition, rule extraction/evaluation, knowledge discovery, data mining.

1 Introduction

Soft *computing* is a consortium of methodologies, which works synergistically and provides flexible information processing capability for handling real life ambiguous situations [1]. Its aim is to exploit the tolerance for imprecision, uncertainty, approximate reasoning and partial truth in order to achieve tractability, robustness, low-cost solutions, and close resemblance to human like decision. The guiding principle is to devise methods of computation which lead to an acceptable solution at low cost by seeking for an approximate solution to an imprecisely/precisely formulated problem.

There are ongoing efforts during the past decade to integrate fuzzy logic, artificial neural networks (ANN) and genetic algorithms (GAs) to build efficient systems in soft computing paradigm. Recently, the theory of rough sets [2], [3] has emerged as another mathematical tool for dealing with uncertainty arising from inexact or incomplete information and is also being used in soft computing [4]. The Rough-fuzzy MLP (Multi Layer Perceptron) [5], developed recently for pattern classification, is such an example combining both rough sets and fuzzy sets with neural networks for building an efficient connectionist system. In this hybridization, fuzzy sets help in handling linguistic input information and ambiguity in output decision, while rough sets extract the domain knowledge for determining the network parameters. Some other attempts in using rough sets (either individually or in combination with fuzzy set) for designing neural network systems are available in [6], [7], [8], [9], where rough sets are used mainly for generating the network parameters and in [10] where roughness at the neuronal level has been incorporated. One may also note the utility of GAs in determining the network parameters as well as the topology (growing/pruning of links), as has been noticed during the past decade [11], [12]. Several algorithms have been developed for extracting embedded knowledge, in the form of symbolic rules, from these hybrid networks [13], [14].

Two important issues which have not been adequately addressed by the above methodologies are those of lengthy training time and poor interpretibility of the networks. A major disadvantage in neural networks learning of large scale tasks is the high computational time required (due to local minima and slow convergence). Use of knowledge-based networks offers only a partial solution to the problem. Also, in most of the above methodologies the link weights of the network are rather uniformly distributed and the network is not suitable for extracting crisp (certain) rules. Compact networks with structure imposed on the weight values are more desirable in this respect for network interpretation. We introduce here the concept of modular learning (in an evolutionary framework) to deal with these problems.

A recent trend in neural network design for large scale problems is to split the original task into simpler subtasks and to coevolve the subnetwork modules for each of the subtasks [15]. The modules are then combined to obtain the final solution. Some of the advantages of this modular approach include decomplexification of the task and its meaningful and clear neural representation. The *divide and conquer* strategy leads to super-linear speedup in training. It also avoids the "temporal crosstalk problem" and interference while learning. In addition, the number of parameters (i.e., weights) can be reduced using modularity; thereby leading to a better generalization performance of the network, compactness in size and crispness in extracted rules.

In this paper, a modular evolutionary approach is adopted for designing a hybrid connectionist system in soft computing framework for both classification and rule

• *The authors are with the Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India.*
E-mail: {sankar, sushmita, pabitra_r}@isical.ac.in.

Manuscript received 8 Mar. 2000; revised 2 May 2001; accepted 9 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 111652.

generation. The basic building block used is the Rough-fuzzy MLP [5], mentioned earlier. The original classification task is split into several subtasks and a number of Rough-fuzzy MLPs are obtained for each subtask. The subnetwork modules are integrated in a particular manner so as to preserve the crude domain knowledge which was encoded in them using rough sets. The pool of integrated networks is then evolved using a GA with a restricted (adaptive/variable) mutation operator that utilizes the domain knowledge to accelerate training and preserves the localized rule structure as potential solutions. The parameters for input and output fuzzy membership functions of the network are also tuned using GA together with the link weights. We have modified the existing procedure for generation of rough set dependency rules for handling directly the real valued attribute table containing fuzzy membership values. This helps in preserving all the class representative points in the dependency rules by adaptively applying a threshold that automatically takes care of the shape of membership functions. Unlike previous attempts of knowledge-based network design [5], [16], [17], here all possible inference rules and not only the best rule, contribute to the final solution. The use of GAs in this context is beneficial for modeling multimodal distributions, since all major representatives in the population are given fair chance during network synthesis.

In the second part of the investigation, a rule extraction algorithm, based on this hybrid model, is presented. The performance of the rules is evaluated quantitatively. Two new measures are accordingly defined indicating the *certainty* and *confusion* in a decision. These new indices are used along with some existing measures to evaluate the quality of the rules. A quantitative comparison of the rule extraction algorithm is made with some existing ones like *Subset* [16], *M of N* [17], and X2R [18] on both real life (speech and medical) and artificially generated data sets with dimensions ranging from two to nine and class boundaries overlapping as well as nonlinear.

The organization of the article is as follows: Section 3 explains, in brief the Rough-fuzzy MLP [5] along with some definitions on rough set theory. Relevant design details of the modular evolutionary algorithms are presented in Section 4. The rule extraction method and the quantitative performance measures are presented in Section 5. Finally, the effectiveness of the proposed model and its comparison with some related ones are provided in Section 5.

2 Rough-fuzzy MLP

The Rough-fuzzy MLP [5] is described briefly in this section. First, we explain the Fuzzy MLP, for convenience. Some definitions on rough set theory are then provided, followed by a discussion on the methodology for extracting rough set dependency rules. Finally, the knowledge encoding algorithm for mapping the rules to the parameters of a fuzzy MLP is explained.

2.1 Fuzzy MLP

The fuzzy MLP model [19] incorporates fuzziness at the input and output levels of the MLP and is capable of handling exact (numerical) and/or inexact (linguistic) forms of input data. Any input feature value is described in terms of some combination of membership values in the linguistic property sets *low* (L), *medium* (M) and *high* (H). Class membership values (μ)) of patterns are represented at the output layer of the fuzzy MLP. During training, the weights are updated by backpropagating errors with respect to these membership values such that the contribution of uncertain vectors is automatically reduced. A three-layered feedforward MLP is used. The output of a neuron in any layer (h) other than the input layer ($h = 0$) is given as

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

where y_i^{h-1} is the state of the ith neuron in the preceding $(h-1)\mathrm{th}$ 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 $x_j^h = \sum_i y_i^{h-1} w_{ji}^{h-1}$. *Input Vector* An n-dimensional pattern $\mathbf{F}_\mathrm{i} = [F_{\mathrm{i}1}, F_{\mathrm{i}2}, \ldots, F_{\mathrm{in}}]$ is represented as a $3n$-dimensional vector

$$\mathbf{F}_\mathrm{i} = [\mu_{low(F_{\mathrm{i}1})}(\mathbf{F}_\mathrm{i}), \ldots, \mu_{high(F_{\mathrm{in}})}(\mathbf{F}_\mathrm{i})] = [\mathrm{y}_1^0, \mathrm{y}_2^0, \ldots, \mathrm{y}_{3\mathrm{n}}^0], \tag{2}$$

where the μ values indicate the membership functions of the corresponding linguistic π-sets *low, medium*, and *high* along each feature axis and $y_1^0, \ldots, y_{3n}^0$ refer to the activations of the $3n$ neurons in the input layer.

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

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

where $\lambda(> 0)$ is the radius of the $\pi\text{-function}$ with c as the central point. Note that features in linguistic and set forms can also be handled in this framework [19].

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

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

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

The membership of the ith pattern in class k, lying in the range $[0, 1]$ is defined as [20]

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

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

2.2 Rough Sets: Definitions

Let us present here some preliminaries of rough set theory, which are relevant to this article. For details one may refer to [2] and [21].

An *information system* is a pair $\mathcal{S} = < U, A >$, where U is a nonempty finite set called the *universe* and A a nonempty finite set of *attributes*. An attribute a can be regarded as a function from the domain U to some value set V_a. A decision system is any information system of the form $\mathcal{A} = (U, A \cup \{d\})$, where $d \not\in A$ is the decision attribute. The elements of A are called conditional attributes.

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

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

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

Then, $I_B = \bigcap_{a \in B} I_a$. If $X \subseteq U$, the sets $\{x \in U : [x]_B \subseteq X\}$ and $\{x \in U : [x]_B \cap X \neq \emptyset\}$, where $[x]_B$ denotes the equivalence class of the object $x \in U$ relative to I_B, are called the *B-lower* and *B-upper approximation* of X in $\mathcal{S}$ and denoted by $\underline{B}X, \overline{B}X$, respectively. $X (\subseteq U)$ is *B-exact* or *B-definable* in $\mathcal{S}$ if $\underline{B}X = \overline{B}X$. It may be observed that $\underline{B}X$ is the greatest B-definable set contained in X and $\overline{B}X$ is the smallest B-definable set containing X.

We now define the notions relevant to knowledge reduction. The aim is to obtain irreducible but essential parts of the knowledge encoded by the given information system; these would constitute *reducts* of the system. So one is, in effect, looking for *maximal* sets of attributes taken from the initial set (A, say), which induce the *same* partition on the domain as A. In other words, the essence of the information remains intact and superfluous attributes are removed. Reducts have been nicely characterized in [21], [22] by *discernibility matrices* and *discernibility functions*. Consider $U = \{x_1, \ldots, x_n\}$ and $A = \{a_1, \ldots, a_m\}$ in the information system $\mathcal{S} = < U, A >$. By the discernibility matrix M $(\mathcal{S})$, of $\mathcal{S}$ is meant an $n \times n$-matrix such that

$$c_{ij} = \{a \in A : a(x_i) \neq a(x_j)\}, \tag{6}$$

A discernibility function $f_{\mathcal{S}}$ is a 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:

$$f_{\mathcal{S}}(\bar{a}_1, \ldots, \bar{a}_m) = \bigwedge\Big\{\bigvee(c_{ij}) : 1 \leq i, j \leq n, j < i, c_{ij} \neq \emptyset\Big\}, \tag{7}$$

where $\bigvee(c_{ij})$ is the disjunction of all variables $\bar{a}$ with $a \in c_{ij}$. It is seen in [21] that $\{a_{i_1}, \ldots, a_{i_p}\}$ is a reduct in $\mathcal{S}$ if and and only if $a_{i_1} \wedge \ldots \wedge a_{i_p}$ is a prime implicant (constituent of the disjunctive normal form) of $f_{\mathcal{S}}$.

2.3 Rough Sets: Dependency Rule Generation

A principal task in the method of rule generation is to compute reducts relative to a particular kind of information system, the decision system. Relativised versions of these matrices and functions shall be the basic tools used in the computation. d-reducts and d-discernibility matrices are used for this purpose [21]. The methodology is described below.

Let $\mathcal{S} = < U, A >$ be a decision table, with C and $D = \{d_1, \ldots, d_l\}$ its sets of condition and decision attributes respectively. Divide the decision table $\mathcal{S} = < U, A >$ into l tables $\mathcal{S}_i = < U_i, A_i >, i = 1, \ldots, l$, corresponding to the l decision attributes $d_1, \ldots, d_l$, where $U = U_1 \cup \ldots \cup U_l$ and $A_i = C \cup \{d_i\}$.

Let $\{x_{i1}, \ldots, x_{ip}\}$ be the set of those objects of U_i that occur in $\mathcal{S}_i, i = 1, \ldots, l$. Now for each d_i-reduct $B = \{b_1, \ldots, b_k\}$ (say), a discernibility matrix (denoted $\mathrm{M}_{d_i}(B)$) from the d_i-discernibility matrix is defined as follows [5]:

$$c_{ij} = \{a \in B : a(x_i) \neq a(x_j)\}, \tag{8}$$

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

For each object $x_j \in x_{i_1}, \ldots, x_{i_p}$, the discernibility function $f_{d_i}^{x_j}$ is defined as

$$f_{d_i}^{x_j} = \bigwedge\Big\{\bigvee(c_{ij}) : 1 \leq i, j \leq n, j < i, c_{ij} \neq \emptyset\Big\}, \tag{9}$$

where $\bigvee(c_{ij})$ is the disjunction of all members of c_{ij}. Then, $f_{d_i}^{x_j}$ is brought to its conjunctive normal form (c.n.f). One thus obtains a dependency rule r_i, *viz.*, $P_i \leftarrow d_i$, where P_i is the disjunctive normal form (d.n.f) of $f_{d_i}^{x_j}, j \in i_1, \ldots, i_p$.

The dependency factor df_i for r_i is given by

$$df_i = \frac{card(POS_i(d_i))}{card(U_i)}, \tag{10}$$

where $POS_i(d_i) = \bigcup_{X \in I_{d_i}} l_i(X)$, and $l_i(X)$ is the lower approximation of X with respect to I_i. In this case, $df_i = 1$ [5].

2.4 Knowledge Encoding

Consider the case of feature F_j for class c_k in the l-class problem domain. The inputs for the ith representative sample $\mathbf{F}_i$ are mapped to the corresponding three-dimensional feature space of $\mu_{low(F_{ij})}(\mathbf{F}_i)$, $\mu_{medium(F_{ij})}(\mathbf{F}_i)$, and $\mu_{high(F_{ij})}(\mathbf{F}_i)$. Let these be represented by L_j, M_j, and H_j respectively. As the method considers multiple objects in a class a separate $n_k \times 3n$-dimensional attribute-value decision table is generated for each class c_k (where n_k indicates the number of objects in c_k).

The absolute distance between each pair of objects is computed along each attribute L_j, M_j, H_j for all j. We modify (8) to directly handle a real-valued attribute table consisting of fuzzy membership values. We define

$$c_{ij} = \{a \in B : \mid a(x_i) - a(x_j) \mid > Th\} \tag{11}$$

for $i, j = 1, \ldots, n_k$, where Th is an adaptive threshold. Note that the adaptivity of this threshold is built in, depending on the inherent shape of the membership function.

Consider Fig. 1. Let a_1, a_2 correspond to two membership functions (attributes) with a_2 being steeper as compared to a_1. It is observed that $r_1 > r_2$. This results in an implicit adaptivity of Th while computing c_{ij} in the discernibility matrix directly from the real-valued attributes. Here lies the novelty of the proposed method. Moreover, this type of thresholding also enables the discernibility matrix to contain all the representative points/clusters present in a class. This is particularly useful in modeling multimodal class distributions.

While designing the initial structure of the Rough-fuzzy MLP, the union of the rules of the l classes is considered. The input layer consists of $3n$ attribute values while the output layer is represented by l classes. The

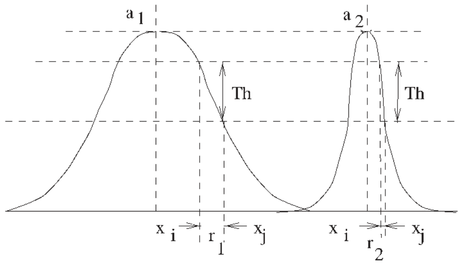


Fig. 1. Illustration of adaptive thresholding of membership functions.

hidden layer nodes model the first level (innermost) operator in the antecedent part of a rule, which can be either a conjunct or a disjunct. The output layer nodes model the outer level operands, which can again be either a conjunct or a disjunct. For each inner level operator, corresponding to one output class (one dependency rule), one hidden node is dedicated. Only those input attributes that appear in this conjunct/disjunct are connected to the appropriate hidden node, which, in turn, is connected to the corresponding output node. Each outer level operator is modeled at the output layer by joining the corresponding hidden nodes. Note that a single attribute (involving no inner level operators) is directly connected to the appropriate output node via a hidden node, to maintain uniformity in rule mapping.

Let the dependency factor for a particular dependency rule for class c_k be $df = \alpha = 1$ by (10). The weight w^1_{ki} between a hidden node i and output node k is set at $\frac{\alpha}{fac} + \varepsilon$, where fac refers to the number of outer level operands 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 $w^0_{ia_j}$ between an attribute a_j (where a corresponds to *low* (L), *medium* (M), or *high* (H)) and hidden node i is set to $\frac{\beta}{facd} + \varepsilon$, such that $facd$ is the number of attributes connected by the corresponding inner level operator. Again $facd \geq 1$. Thus, for an l-class problem domain, there are at least l hidden nodes. It is to be mentioned that the number of hidden nodes is determined directly from the dependency rules. It depends on the form in which the antecedents are present in the rules.

3 Modular Evolution of Rough-fuzzy MLP

The design procedure of Modular Neural Networks (MNN) involves two broad steps—effective decomposition of the problem such that the subproblems can be solved with compact networks and efficient combination and training of the networks such that there is gain in terms of training time, network size and accuracy. These are described in detail in the following section along with the steps involved and the characteristics features.

3.1 Algorithm

We use two phases. First, an l-class classification problem is split into l two-class problems. Let there be l sets of subnetworks, with $3n$ inputs and one output node each. Rough set theoretic concepts are used to encode domain knowledge into each of the subnetworks, using (9), (10), and (11). As explained in Section 2.4, the number of hidden nodes and connectivity of the knowledge-based subnetworks is automatically determined. Each two-class problem leads to the generation of one or more crude subnetworks, each encoding a particular decision rule. Let each of these constitute a pool. So, we obtain $m \geq l$ pools of knowledge-based modules. Each pool k is perturbed to generate a total of n_k subnetworks, such that $n_1 = \ldots = n_k = \ldots = n_m$. These pools constitute the initial population of subnetworks, which are then evolved independently using genetic algorithms.

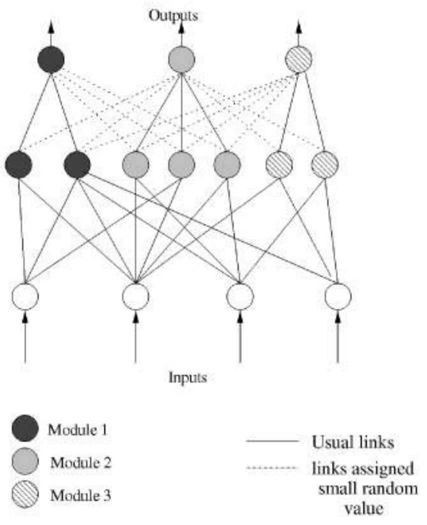


Fig. 2. Intra and intermodule links.

At the end of the above phase, the modules/subnetworks corresponding to each two-class problem are concatenated to form an initial network for the second phase. The inter module links are initialized to small random values as depicted in Fig. 2. A set of such concatenated networks forms the initial population of the GA. The mutation probability for the intermodule links is now set to a high value, while that of intramodule links is set to a relatively lower value. This sort of *restricted* mutation helps preserve some of the localized rule structures, already extracted and evolved, as potential solutions. The initial population for the GA of the entire network is formed from all possible combinations of these individual network modules and random perturbations about them. This ensures that for complex multimodal pattern distributions all the different representative points remain in the population. The algorithm then searches through the reduced space of possible network topologies. The steps are summarized below followed by an example.

3.1.1 Steps

Step 1. For each class, generate rough set dependency rules using the methodology described in Section 2.3.

Step 2. Map each of the dependency ru1les to a separate subnetwork modules (Fuzzy MLPs) using the methodology described in Section 2.4.

Step 3. Partially evolve each of the subnetworks using conventional GA.

Step 4. Concatenate the subnetwork modules to obtain the complete network. For concatenation the intramodule links

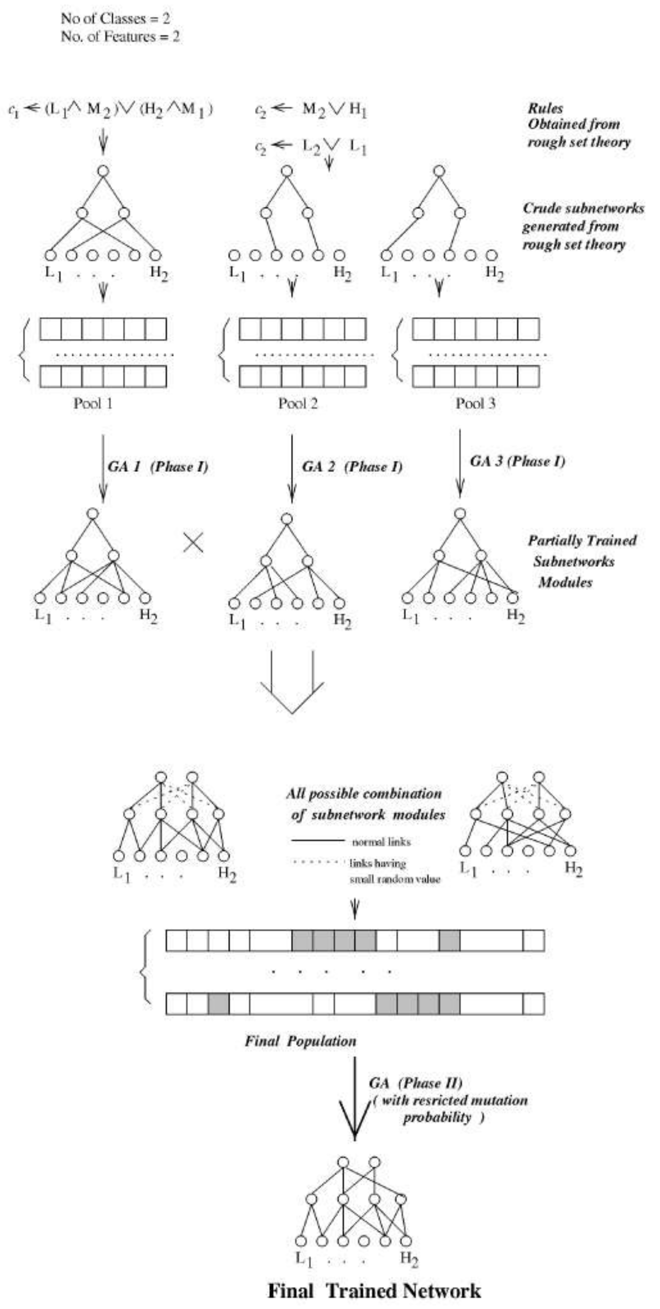

No of Classes = 2
No. of Features = 2
c1 ← (L1 ∧ M2) ∨ (H2 ∧ M1)
c2 ← M2 ∨ H1
c2 ← L2 ∨ L1
Rules Obtained from rough set theory
Crude subnetworks generated from rough set theory
L1 . . . H2
Pool 1
Pool 2
Pool 3
GA 1 (Phase I)
GA 2 (Phase I)
GA 3 (Phase I)
Partially Trained Subnetworks Modules
All possible combination of subnetwork modules
normal links
links having small random value
Final Population
GA (Phase II) (with resricted mutation probability)
Final Trained Network

Fig. 3. Steps for designing a sample Modular Rough-fuzzy MLP.

are left unchanged while the intermodule links are initialized to low random values. Note that each of the subnetworks solves a 2-class classification problem, while the concatenated network solve the actual l-class problem. Every possible combination of subnetwork modules is generated to form a pool of networks.

Step 5. The pool of networks is evolved using a *modified* GA with an adaptive/variable mutation operator. The mutation probability is set to a low value for the intramodule links and to a high value for the intermodule links.

3.1.2 Example

Consider a problem of classifying a two dimensional data into two classes. The input fuzzifier maps the features into a six-dimensional feature space. Let a sample set of rules obtained from rough set theory be

$$c_1 \leftarrow (L_1 \wedge M_2) \vee (H_2 \wedge M_1), c_2 \leftarrow M_2 \vee H_1, c_2 \leftarrow L_2 \vee L_1,$$

where L_j, M_j, H_j correspond to $\mu_{low(F_j)}$, $\mu_{medium(F_j)}$, $\mu_{high(F_j)}$, respectively. For the first phase of the GA three different pools are formed, using one crude subnetwork for class 1 and two crude subnetworks for class 2, respectively. Three partially trained subnetworks result from each of these pools. They are then concatenated to form $(1 \times 2) = 2$ networks. The population for the final phase of the GA is formed with these networks and perturbations about them. The steps followed in obtaining the final network is illustrated in Fig. 3. Note that Fig. 4 explains the chromosome representation of intra and intermodule links.

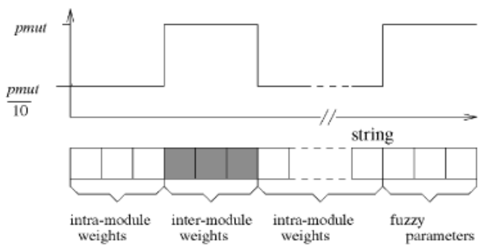


Fig. 4. Variation of mutation probability along the encoded string.

3.1.3 Characteristic Features

1. The use of rough sets for knowledge encoding provides an established mathematical framework for network decomposition. Knowledge encoding not only produces an initial network close to the optimal one, it also reduces the search space. The initial network topology is automatically determined and provides good *building blocks* for the GA.
2. In earlier concurrent algorithms for neural network learning, there exist no guidelines for the decomposition of network modules in [23]. Arbitrary subnetworks are assigned to each of the classes. Use of networks with the same number of hidden nodes for all classes leads to overlearning in the case of simple classes and poor learning in complex classes. Use of rough set theory circumvents the above problem.
3. Sufficient reduction in training time is obtained, as the above approach parallelizes the GA to an extent. The search string for the GA for subnetworks being smaller, more than linear decrease in searching time is obtained. Also very small number of training cycles are required in the refinement phase, as the network is already very close to the solution. Note that the modular aspect of our algorithm is similar to the coevolutionary algorithm (CEA) used for solving large scale problems with EAs [23].
4. The splitting of an l-class problem into l two-class problems bears analogy to the well known *divide and conquer* strategy and speeds up the search procedure significantly. Here, one can use a smaller chromosome and/or population size, thereby alleviating to some extent the space-time complexity problem.

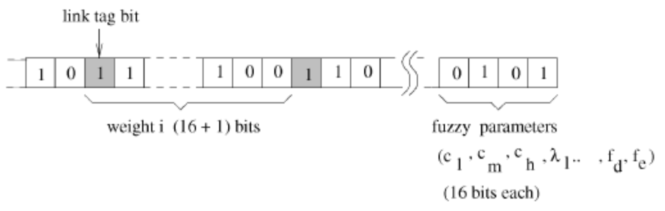
5. The algorithm indirectly constrains the solution in such a manner that a structure is imposed on the connection weights. This is helpful for subsequent rule-extraction from the weights, as the resultant network obtained has sparse but strong interconnection among the nodes. Although in the above process some amount of optimality is sacrificed and often for many-class problems the number of nodes required may be higher than optimal, yet the network is less redundant. However the nature of objective function considered and the modular knowledge-based methodology used enables sufficient amount of link pruning and the total number of links are found to be significantly less. The use of *restricted* mutation (as defined in Section 3.2.3) minimizes the destruction of encoded rule structures in the knowledge-based networks.
6. For each two-class (sub)problem a set of subnetworks encoding separate decision rules are available. Since all possible combination of these subnetworks are considered for the final evolutionary training, greater diversity within the population is possible. This results in faster convergence of the GA which utilizes multiple theories about a domain. This also ensures that all the clusters in the feature space are adequately represented in the final solution.

3.2 Evolutionary Design

Here, we discuss different features of the genetic algorithm [24] with relevance to our algorithm.

3.2.1 Chromosomal Representation

The problem variables consist of the weight values and the input/output fuzzification parameters. Each of the weights is encoded into a binary word of 16 bit length, where $[000\ldots0]$ decodes to -128 and $[111\ldots1]$ decodes to 128. An additional bit is assigned to each weight to indicate the presence or absence of the link. The fuzzification parameters tuned are the centers (c) and radius (λ) for each of the linguistic attributes *low*, *medium*, and *high* of each feature, and the output fuzzifiers f_d and f_e [19]. These are also coded as 16 bit strings in the range $[0, 2]$. For the input parameters, $[000\ldots0]$ decodes to 0 and $[111\ldots1]$ decodes to 1.2 times the maximum value attained by the corresponding feature in the training set. The chromosome is obtained by concatenating all the above strings. Sample values of the string length are around 2000 bits for reasonably sized networks.



Initial population is generated by coding the networks obtained by rough set-based knowledge encoding and by random perturbations about them. A population size of 64 was considered.

3.2.2 Crossover

It is obvious that due to the large string length, single point crossover would have little effectiveness. Multiple point crossover is adopted, with the distance between two crossover points being a random variable between eight and 24 bits. This is done to ensure a high probability for only one crossover point occurring within a word encoding a single weight. The crossover probability is fixed at 0.7.

3.2.3 Mutation

The search string being very large, the influence of mutation is more on the search compared to crossover. The mutation probability has a spatio-temporal variation. The maximum value of $pmut$ is chosen to be 0.4 and the minimum value as 0.01. The mutation probabilities also vary along the encoded string, the bits corresponding to intermodule links being assigned a probability $pmut$ (i.e., the value of $pmut$ at that iteration) and intramodule links assigned a probability $pmut/10$. This is done to ensure least alterations in the structure of the individual modules already evolved. Hence, the mutation operator indirectly incorporates the domain knowledge extracted through rough set theory.

3.2.4 Choice of Fitness Function

An objective function of the form described below is chosen.

$$F = \alpha_1 f_1 + \alpha_2 f_2, \tag{12}$$

where

$$f_1 = \frac{No.\ of\ Correctly\ Classified\ Sample\ in\ Training\ Set}{Total\ No.\ of\ Samples\ in\ Training\ Set}$$

$$f_2 = 1 - \frac{No.\ of\ links\ present}{Total\ No.\ of\ links\ possible}.$$

Here, α_1 and α_2 determine the relative weight of each of the factors. α_1 is taken to be 0.9 and α_2 is taken as 0.1, to give more importance to the classification score compared to the network size in terms of number of links. Note that we optimize the network connectivity, weights and input/output fuzzification parameters simultaneously.

Selection is done by the *roulette wheel* method. The probabilities are calculated on the basis of ranking of the individuals in terms of the objective function, instead of the objective function itself. *Elitism* is incorporated in the selection process to prevent oscillation of the fitness function with generation. The fitness of the best individual of a new generation is compared with that of the current generation. If the latter has a higher value the corresponding individual replaces a randomly selected individual in the new population.

4 Rule Generation and Quantitative Evaluation

4.1 Extraction Methodology

Algorithms for rule generation from neural networks mainly fall in two categories—pedagogical and decompositional [13]. Our algorithm can be categorized as decompositional. It is described below.

1. Compute the following quantities: $PMean = Mean$ of all positive weights, $PThres_1 = \text{Mean}$ of all positive weights less than $PMean$, $PThres_2 = \text{Mean}$ of all weights greater than $PMean$. Similarly calculate $NThres_1$ and $NThres_2$ for negative weights.
2. For each hidden and output unit
   a. for all weights greater than $PThres_1$ search for positive rules only and for all weights less than $NThres_1$ search for negated rules only by *Subset* method.
   b. search for combinations of positive weights above $Pthres_2$ and negative weights greater than $NThres_1$ that exceed the bias. Similarly search for negative weights less than $NThres_2$ and positive weights below $PThres_1$ to find out rules.
3. Associate with each rule j a confidence factor

$$cf_j = \inf_{j:\text{all nodes in the path}} \frac{(\Sigma_i w_{ji} - \theta_j)}{\Sigma_i w_{ji}}, \qquad (13)$$

where w_{ji} is the ith incoming link weight to node j.

Since our training algorithm imposes a structure on the network, resulting in a sparse network having few strong links, the $PThres$ and $NThres$ values are well separated. Hence, the above rule extraction algorithm generates most of the embedded rules over a small number of computational steps.

The computational complexity of our algorithm is as follows. Let the network have i, h, o numbers of input, hidden and output nodes, respectively. Let us make the assumption that $i = h = o = k$. Let the fraction of weights having value in $[0, PThres_1)$, $[PThres_1, PThres_2)$, $[PThres_2, \infty)$, be p_1, p_2, p_3, respectively. Similarly let the corresponding fractions for negative weights be n_1, n_2, n_3. Then, the computational complexity ($\mathcal{C}$) becomes

$$\mathcal{C} = k.(2^{(p_2+p_3)k+1} + 2^{(n_2+n_3)k+1} + 2^{(p_3+n_1)k+1} + 2^{(p_1+n_3)k+1}).$$

If $n_1, n_2, p_1, p_2 \ll p_3, n_3$,

$$\mathcal{C} \approx 4k.(2^{p_3k} + 2^{n_3k}) = 4k.(e^l n2.p_3k + e^l n2.n_3k).$$

Also if $p_3, n_3 \ll 1$,

$$\mathcal{C} \approx 4k.(1 + ln2.(p_3 + n_3)k + 0.5.(ln2.(p_3 + n_3))^2 k^2,$$

i.e., $\mathcal{C} \approx \mathcal{O}(k^3)$.

An important consideration is the order of application of rules in a rule base. Since most of the real life patterns are noisy and overlapping, rule bases obtained are often not totally consistent. Hence, multiple rules may fire for a single example. Several existing approaches apply the rules sequentially [25], often leading to degraded performance. The rules extracted by our method have confidence factors associated with them. Therefore, if multiple rules are fired we use the strongest rule having the highest confidence.

Two existing rule extraction algorithms, similar in spirit to the proposed algorithm, are the *Subset* method [16] and *M of N* method [17]. The major problem with the *Subset* algorithm is that the cost of finding all subsets grows as the size of the power set of the links to each unit. It requires lengthy, exhaustive searches of size $\mathcal{O}(2^k)$ for a hidden/output node with a fan-in of k and extracts a large set of rules, upto $\beta_p * (1 + \beta_n)$, where β_p and β_n are the number of subsets of positively and negatively weighted links respectively. Some of the generated rules may be repetitive, as permutations of rule antecedents are not taken care of automatically. Moreover, there is no guarantee that all useful knowledge embedded in the trained network will be extracted. Computational complexity of the *M of N* algorithm is $\mathcal{O}(k^3 + (k^2.j))$, where j is the number of examples. Additionally, the rule extraction procedure involves a backpropagation step requiring significant computation time. The algorithm has good generalization (accuracy), but can have degraded comprehensibility [26]. Note that one considers groups of links as equivalence classes, thereby generating a bound on the number of rules rather than establishing a ceiling on the number of antecedents.

4.2 Quantitative Measures

Here, we provide some measures in order to evaluate the performance of the rules. Among them *Certainty* and *Confusion* reflecting the confidence and ambiguity in a decision, are newly defined. Note that these aspects had not been considered earlier.

Let N be an $l \times l$ matrix whose (i, j)th element n_{ij} indicate the number of patterns actually belonging to class i, but classified as class j.

1. *Accuracy.* It is the correct classification percentage, provided by the rules on a test set defined as $\frac{n_{ic}}{n_i}.100$, where n_i is equal to the number of points in class i and n_{ic} of these points are correctly classified.
2. *User's Accuracy* [27]. If n'_i points are found to be classified into class i, then the user's accuracy (U) is defined as $U = n_{ic}/n'_i$. This gives a measure of the confidence that a classifier attributes to a region as belonging to a class. In other words, it denotes the level of purity associated with a region.
3. *Kappa* [27]. The coefficient of agreement called "kappa" measures the relationship of beyond chance agreement to expected disagreement. It uses all the cells in the confusion matrix, not just the diagonal elements. The estimate of kappa (K) is the proportion of agreement after chance agreement is removed from consideration. The kappa value for class i (K_i) is defined as

$$K_i = \frac{n.n_{ic} - n_i.n'_i}{n.n'_i - n_i.n'_i}. \qquad (14)$$

   The numerator and denominator of overall kappa are obtained by summing the respective numerators and denominators of K_i separately over all classes.
4. *Fidelity* [26]. This represents how closely the rule base approximates the parent neural network model [26]. We measure this as the percentage of the test set for which network and the rule base output agree. Note that fidelity may or may not be greater than accuracy.
5. *Confusion.* This measure quantifies the goal that the "*Confusion should be restricted within minimum number of classes.*" This property is helpful in higher level decision making. Let $\hat{n}_{ij}$ be the mean of all n_{ij} for $i \neq j$. Then, we define

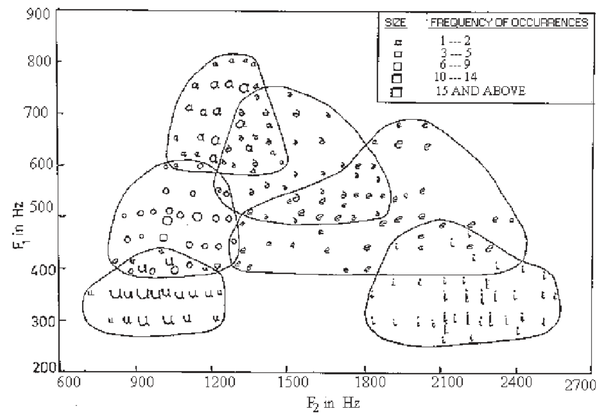


Fig. 5. Projection in $F_1 - F_2$ plane of the **Vowel** data.

$$Conf = \frac{Card\{n_{ij} : n_{ij} \geq \hat{n}_{ij}, i \neq j\}}{l} \quad (15)$$

for an l class problem. The lower the value of $Conf$, lesser is the number of classes between which confusion is occurs.

6. *Cover.* Ideally the rules extracted should cover all the cluster regions of the pattern space. We use the percentage of examples from a test set for which no rules are fired as a measure of the uncovered region. A rule base having a smaller uncovered region is superior.
7. *Rule base size.* It is measured in terms of the number of rules. Lower the value is, the more compact is the rule base.
8. *Computational complexity.* Here, we present the CPU time required.
9. *Certainty.* By certainty of a rule base, we quantify the confidence of the rules as defined by the certainty factor cf (13).

5 Implementation and Results

The genetic-rough-neuro-fuzzy algorithm has been implemented on both real life (speech, medical) and artificially generated data. The data sets are available at http://www.isical.ac.in/~sushmita/patterns. Let the proposed methodology be termed Model S. Other models compared include:

Model O: An ordinary MLP trained using backpropagation (BP) with weight decay.

Model F: A fuzzy multilayer perceptron trained using BP [19] (with weight decay).

Model R: A fuzzy multilayer perceptron trained using BP (with weight decay), with initial knowledge encoding using rough sets [5].

Model FM.: A modular fuzzy multilayer perceptron trained with GAs along with tuning of the fuzzification parameters. Here, the term modular refers to the use of subnetworks corresponding to each class, that are later concatenated using GAs.

The speech data **Vowel** deals with 871 Indian Telegu 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 set has three features: F_1, F_2, and F_3 corresponding to the first, second and third vowel formant frequencies obtained through spectrum analysis of the speech data. Fig. 5 depicts the projection in the $F_1 - F_2$ plane, of the six vowel classes δ, a, i, u, e, o. These overlapping classes will be denoted by $c_1, c_2, \ldots, c_6$.

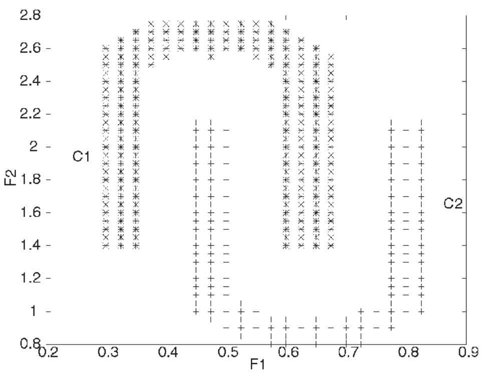


Fig. 6. Artificially generated linearly nonseperable pattern **Pat**.

The synthetic data **Pat** consists of 880 pattern points in the two-dimensional space $F_1 - F_2$, as depicted in Fig. 6. There are three linearly nonseparable pattern classes. The figure is marked with classes 1 (c_1) and 2 (c_2), while class 3 (c_3) corresponds to the background region.

The medical data consisting of nine input features and four pattern classes, deals with various *Hepatobiliary disorders* 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 classes. These are referred to as c_1, c_2, c_3, c_4.

TABLE 1
Rough Set Dependency Rules for **Vowel** Data along with the Input Fuzzification Parameter Values

c_1	$\leftarrow$	$M_1 \vee L_3$
c_1	$\leftarrow$	$M_1 \vee M_2$
c_2	$\leftarrow$	$M_2 \vee M_3 \vee (H_1 \wedge M_2)$
c_2	$\leftarrow$	$M_2 \vee H_3$
c_3	$\leftarrow$	$(L_1 \wedge H_2) \vee (M_1 \wedge H_2)$
c_3	$\leftarrow$	$(L_1 \wedge H_2) \vee (L_1 \wedge M_3)$
c_4	$\leftarrow$	$(L_1 \wedge L_2) \vee (L_1 \wedge L_3) \vee (L_2 \wedge M_3) \vee (L_1 \wedge M_3)$
c_5	$\leftarrow$	$(H_1 \wedge M_2) \vee (M_1 \wedge M_3) \vee (M_1 \wedge M_2) \vee (M_2 \wedge L_1)$
c_5	$\leftarrow$	$(H_1 \wedge M_2) \vee (M_1 \wedge M_2) \vee (H_1 \wedge H_3) \vee (H_2 \wedge L_1)$
c_5	$\leftarrow$	$(L_2 \wedge L_1) \vee (H_3 \wedge M_3) \vee M_1$
c_6	$\leftarrow$	$L_1 \vee M_3 \vee L_2$
c_6	$\leftarrow$	$M_1 \vee H_3$
c_6	$\leftarrow$	$L_1 \vee H_3$
c_6	$\leftarrow$	$M_1 \vee M_3 \vee L_2$.

Fuzzification Parameters:

Feature 1: $c_L = 0.348$, $c_M = 0.463$, $c_H = 0.613$, $\lambda_L = 0.115$, $\lambda_M = 0.150$, $\lambda_H = 0.134$

Feature 2: $c_L = 0.219$, $c_M = 0.437$, $c_H = 0.725$, $\lambda_L = 0.218$, $\lambda_M = 0.253$, $\lambda_H = 0.288$

Feature 3: $c_L = 0.396$, $c_M = 0.542$, $c_H = 0.678$, $\lambda_L = 0.146$, $\lambda_M = 0.140$, $\lambda_H = 0.135$

TABLE 2
Comparative Performance of Different Models

Models	Model O		Model F		Model R		Model FM		Model S	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Vowel data										
Accuracy(%) (Mean SD)	65.4 0.5	64.1 0.5	84.1 0.4	81.8 0.5	86.7 0.3	86.0 0.2	85.3 0.4	82.3 0.5	87.1 0.2	85.8 0.2
# links	131		210		152		124		84	
Sweeps	5600		5600		2000		200		90	
Pat data										
Accuracy(%) (Mean SD)	55.1 0.4	54.8 0.3	68.7 0.5	68.1 0.5	73.1 0.4	71.1 0.4	70.2 0.5	69.8 0.4	75.7 0.5	74.7 0.4
# links	62		105		82		84		72	
Sweeps	2000		2000		1500		150		90	
Medical data										
Accuracy(%) (Mean SD)	70.1 0.4	60.0 0.3	66.1 0.4	69.8 0.5	76.9 0.4	68.0 0.5	76.8 0.4	67.4 0.5	78.4 0.4	68.9 0.5
# links	143		310		190		230		108	
Iterations	2500		2500		1500		200		110	

SD: Standard Deviation

5.1 Classification

Recognition scores obtained for each of the data by the proposed soft modular network (Model S) are presented in Table 2. It also shows a comparison with other related MLP-based classification methods (Models O, F, R and FM). In all cases, 10 percent of the samples are used as training set and the remaining samples are used as test set. Ten such independent runs are performed and the mean value and standard deviation of the classification accuracy, computed over them, are presented in Table 2.

The dependency rules, as generated via rough set theory and used in the encoding scheme, are shown in Table 1 only for vowel data, as an example. The values of input fuzzification parameters used are also presented in Table 1. The corresponding π-functions are shown in Fig. 7 only for feature F_1, as an illustration. In Table 1, F_i, where F stands for *low*, *medium*, or *high*, denotes a property F of the ith feature [19]. The integrated networks contain 18, 15, and 10 hidden nodes in a single layer for *Vowel*, *Pat*, and medical data, respectively. After combination 96, 61, and 16 networks were obtained, respectively. The initial population of the GA was formed using 64 networks in each of these cases. In the first phase of the GA (for models FM and S), each of the subnetworks are partially trained for 10 sweeps.

The classification accuracies obtained by the models are analysed for statistical significance. Tests of significance are performed for the inequality of means (of accuracies) obtained using the proposed algorithm and the other methods compared. Since both mean pairs and the variance pairs are unknown and different, a generalized version of t-test is appropiate in this context. This problem is the classical Behrens-Fisher problem in

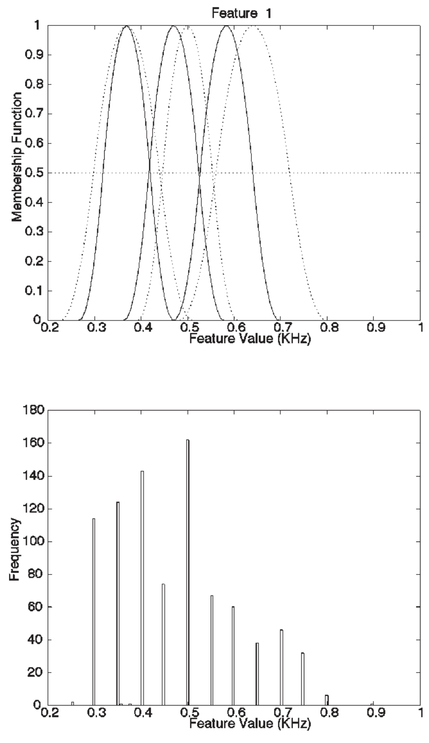


Fig. 7. Input π-functions and data distribution along F_1 axis for the Vowel data. Solid lines represent the initial functions and dashed lines represent the functions obtained finally after tuning with GAs. The horizontal dotted lines represents the threshold level.

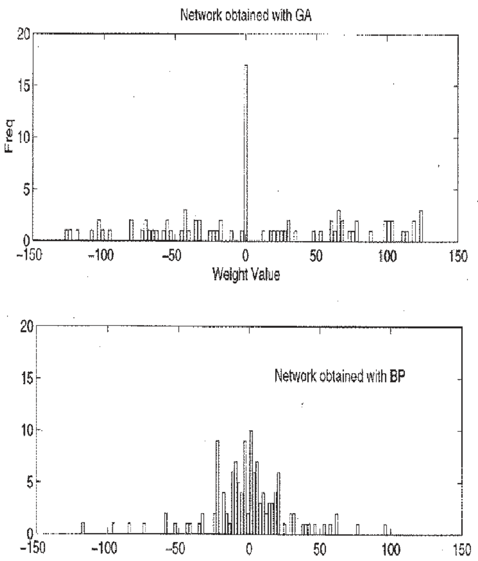


Fig. 8. Histogram plot of the distribution of weight values with Model S and Model F for **Vowel** data.

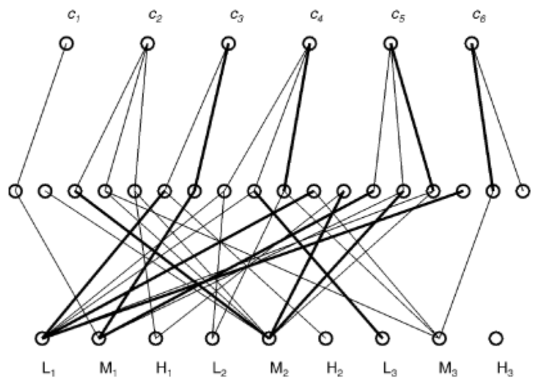


Fig. 9. Positive connectivity of the network obtained for the **Vowel** data, using Model S. (Bold lines indicate weights greater than $PThres_2$, while others indicate values between $PThres_1$ and $PThres_2$).

hypothesis testing, a suitable test statistic is described in [28] and tabled in [29]. The test statistic is of the form

$$v = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\lambda_1 s_1^2 + \lambda_2 s_2^2}},$$

where $\bar{x}_1, \bar{x}_2$ are the means, s_1, s_2 are the standard deviations, $\lambda_1 = 1/n_1$, $\lambda_2 = 1/n_2$, and n_1, n_2 are the number of observations. Since, experiments were performed on 10 independent random training sets for all the algorithms, we have $n_1 = n_2 = 10$. The test confidence level considered was 95 percent. In Table 2, we present the mean and standard deviation (SD) of the accuracies. Using the means and SDs, the value of the test statistics is computed. If the value exceeds the corresponding tabled value, the means are unequal with statistical significance (algorithm having higher mean accuracy being significantly superior to the one having lower value).

It is observed from Table 2 that Model S performs the best (except for Model R on Vowel data and Model F on Medical data) with the least network size as well as least number of sweeps. For Model R with Vowel data and Model F with Medical data, the classification performance on test set is marginally better than that of Model S, but with significantly higher number of links and training sweeps required. Comparing models F and R, we observe that the incorporation of domain knowledge in the latter through rough sets boosts its performance. Similarly, using the modular approach with GA (Model FM) improves the efficiency of Model F. Since Model S encompasses the principle of both models R and FM, it results in the least redundant yet most effective model. The variation of the classification accuracy of the models with iteration is also studied. As expected, Model S is found to have high recognition score at the very begining of evolutionary training, the next values are attained by models R and FM, and the lowest being attained by models O and F using backpropagation. For example, in the case of Vowel data, these figures are 64 percent for S, 52 percent for R, 44 percent for FM, and 0 percent for F and O. Model S converges after about 90 iterations of the GA, providing the highest accuracy compared to all the other models. The backpropagation-based models require about 2,000-5,000 iterations for convergence.

It may be noted that the training algorithm suggested is successful in imposing a structure among the connection weights. As seen from Fig. 8, for vowel data, the weight values for a fuzzy MLP trained with BP (Model F) is more or less uniformly distributed between the maximum and minimum values. On the other hand, the Modular Rough-fuzzy MLP (Model S) has most of its weight values zero while majority of its nonzero weights have a high value. Hence, it can be inferred that the former model results in a dense network with weak links, while the incorporation of rough sets, modular concepts and GAs produces a sparse network with strong links. The latter is suitable for rule extraction. The connectivity (positive weights) of the trained network is shown in Fig. 9.

5.2 Rule Extraction

We use the algorithm explained in Section 4.1 to extract rules from the trained network (Model S). These rules are compared with those obtained by the *Subset* method [16], *M of N* method [17], a pedagogical method X2R [18], and a decision tree-based method C4.5 [30] in terms of the performance measures (Section 4.2). The set of rules extracted from the proposed network (Model S) is presented in Table 4 along with their certainty factors (*cf*). The values of the fuzzification parameters of the membership functions L, M, and H are also mentioned. For the medical data, we present the fuzzification parameters only for those features that appear in the extracted rules.

TABLE 3
Comparison of the Performance of the Rules Extracted by Various Methods for **Vowel**, **Pat**, and **Medical** Data

	Algorithm	Accuracy (%)	Users' Accuracy (%)	Kappa (%)	Uncovered Region (%)	No. of Rules	CPU time (Sec)	$Conf$
V	Model S	81.02	83.31	78.17	3.10	10	1.1	1.4
O	Subset	82.01	82.72	77.29	2.89	16	1.4	1.9
W	M of N	79.00	80.01	74.55	2.10	14	1.2	1.9
E	X2R	76.00	75.81	72.34	2.72	14	0.9	1.7
L	C4.5	79.00	79.17	77.21	3.10	16	1.0	1.5
	Model S	70.31	74.44	71.80	2.02	8	1.0	1.1
P	Subset	71.02	73.01	70.09	1.91	16	1.1	1.5
A	M of N	70.09	71.12	70.02	2.02	11	1.1	1.5
T	X2R	67.82	68.23	67.91	1.91	10	0.9	1.4
	C4.5	71.02	73.44	72.00	2.02	11	1.1	1.2
M	Model S	64.90	64.70	64.10	8.02	7	0.9	1.4
E	Subset	65.00	65.41	64.44	7.52	11	1.0	1.8
D	M of N	63.91	64.00	63.02	8.02	10	1.0	1.8
C	X2R	61.02	60.90	60.90	7.91	9	0.9	1.7
AL	C4.5	64.01	64.23	64.90	7.91	10	0.9	1.4

TABLE 4
Rules Extracted from Trained Networks (Model S) for **Vowel**, **Pat**, and **Medical** Data along with the Input Fuzzification Parameter Values

Vowel data

c_1	$\leftarrow$	$M_1 \vee L_3 \vee M_2$	$cf = 0.851$
c_1	$\leftarrow$	$H_1 \vee M_2$	$cf = 0.755$
c_2	$\leftarrow$	$M_2 \vee M_3$	$cf = 0.811$
c_2	$\leftarrow$	$\neg M_1 \wedge \neg H_1 \wedge L_2 \wedge M_2$	$cf = 0.846$
c_3	$\leftarrow$	$L_1 \vee H_2$	$cf = 0.778$
c_4	$\leftarrow$	$L_1 \wedge L_2 \wedge \neg L_3$	$cf = 0.719$
c_5	$\leftarrow$	$M_1 \wedge H_2$	$cf = 0.881$
c_5	$\leftarrow$	$M_1 \wedge M_2$	$cf = 0.782$
c_5	$\leftarrow$	$H_1 \wedge M_2$	$cf = 0.721$
c_6	$\leftarrow$	$\neg H_2$	$cf = 0.717.$

$FuzzificationParameters:$

$Feature\ 1:$	$c_L = 0.34,$	$c_M = 0.502,$	$c_H = 0.681$
$Feature\ 1:$	$\lambda_L = 0.122,$	$\lambda_M = 0.154,$	$\lambda_H = 0.177$
$Feature\ 2:$	$c_L = 0.217,$	$c_M = 0.431,$	$c_H = 0.725$
$Feature\ 2:$	$\lambda_L = 0.211,$	$\lambda_M = 0.250,$	$\lambda_H = 0.288$
$Feature\ 3:$	$c_L = 0.380,$	$c_M = 0.540,$	$c_H = 0.675$
$Feature\ 3:$	$\lambda_L = 0.244,$	$\lambda_M = 0.212,$	$\lambda_H = 0.224$

Pat data

c_1	$\leftarrow$	$M_1 \wedge M_2$	$cf = 0.674$
c_1	$\leftarrow$	$M_1 \wedge H_1 \wedge \neg L_2$	$cf = 0.875$
c_2	$\leftarrow$	$L_2 \wedge H_1$	$cf = 0.80$
c_2	$\leftarrow$	$L_2 \wedge M_1$	$cf = 0.778$
c_3	$\leftarrow$	$L_1 \wedge L_2$	$cf = 0.636$
c_3	$\leftarrow$	$H_1 \wedge H_2$	$cf = 0.674$
c_3	$\leftarrow$	$M_1 \wedge M_2 \wedge \neg L_2$	$cf = 0.636$
c_3	$\leftarrow$	$M_1 \wedge M_2 \wedge \neg H_2$	$cf = 0.636.$

$FuzzificationParameters:$

$Feature\ 1:$	$c_L = 0.216$	$c_M = 0.499$	$c_H = 0.751$
$Feature\ 1:$	$\lambda_L = 0.282$	$\lambda_M = 0.265$	$\lambda_H = 0.252$
$Feature\ 2:$	$c_L = 1.266$	$c_M = 1.737$	$c_H = 2.511$
$Feature\ 2:$	$\lambda_L = 0.244$	$\lambda_M = 0.235$	$\lambda_H = 0.226$

Medical data

c_1	$\leftarrow$	$L_3 \wedge M_6 \wedge \neg L_1$	$cf = 0.857$
c_1	$\leftarrow$	$L_3 \wedge \neg L_6 \wedge L_1$	$cf = 0.800$
c_2	$\leftarrow$	$L_2 \wedge M_2 \wedge \not M_3$	$cf = 0.846$
c_2	$\leftarrow$	M_6	$cf = 0.571$
c_3	$\leftarrow$	$L_3 \wedge L_2 \wedge M_3$	$cf = 0.800$
c_4	$\leftarrow$	$L_3 \wedge L_6 \wedge \neg L_1$	$cf = 0.833$
c_4	$\leftarrow$	$M_2 \wedge L_2 \wedge \neg M_3$	$cf = 0.833$

$FuzzificationParameters:$

$Feature\ 1:$	$c_L = 52.47$	$c_M = 112.88$	$c_H = 289.17$
$Feature\ 1:$	$\lambda_L = 62.44$	$\lambda_M = 118.35$	$\lambda_H = 176.40$
$Feature\ 2:$	$c_L = 24.04$	$c_M = 53.41$	$c_H = 125.35$
$Feature\ 2:$	$\lambda_L = 29.15$	$\lambda_M = 55.14$	$\lambda_H = 72.08$
$Feature\ 3:$	$c_L = 336.15$	$c_M = 477.95$	$c_H = 844.00$
$Feature\ 3:$	$\lambda_L = 140.00$	$\lambda_M = 254.44$	$\lambda_H = 367.01$
$Feature\ 6:$	$c_L = 12.15$	$c_M = 17.27$	$c_H = 25.52$
$Feature\ 6:$	$\lambda_L = 5.11$	$\lambda_M = 7.18$	$\lambda_H = 9.25$

A comparison of the performance indices of the extracted rules is presented in Table 3. Since the network obtained using Model S contains fewer links, the generated rules are less in number and they have high *certainty factor*. Accordingly, it possesses relatively higher percentage of uncovered region, though the accuracy did not suffer much. Although the *Subset* algorithm achieves the highest *accuracy*, it requires the largest number of rules and computation time. In fact, the accuracy/computation time of Subset method is marginally better/worse than Model S, while the size of the rule base is significantly less for Model S.

The accuracy achieved by Model S is better than that of *M of N*, X2R, and C4.5, except for the *Pat* data with C4.5. Also, considering *user's accuracy* and *kappa*, the best performance is obtained by Model S. The X2R algorithm requires least computation time but achieves the least accuracy with more rules. The *Conf* index is the minimum for rules extracted by Model S; it also has high *fidelity* (e.g., 94.22 percent, 89.17 percent and 74.88 percent for *Vowel*, *Pat*, and medical data respectively).

In a part of the experiment, we also conducted a comparison with Models F, R, and FM for rule extraction. It was observed that the performance degrades substantially for them because these networks are less structured and hence less suitable, as compared to Model S, for rule extraction.

6 Conclusions and Discussion

A methodology for modular evolution of a rough-fuzzy-MLP using genetic algorithms for designing a knowledge-based network for pattern classification and rule generation is presented. The proposed algorithm involves synthesis of several MLP modules, each encoding the rough set rules for a particular class. These knowledge-based modules are refined using a GA. The genetic operators are implemented in such a way that they help preserve the modular structure already evolved. It is seen that this methodology along with modular network decomposition results in accelerated training and more sparse (compact) network with comparable classification accuracy, as compared to earlier hybridizations.

The aforesaid model is used to develop a new rule extraction algorithm. The extracted rules are compared with some of the related rule extraction techniques on the basis of some quantitative performance indices. Two new measures, introduced to evaluate the confidence and ambiguity in a decision, are found to be satisfactory. It is observed that the proposed methodology extracts rules which are less in number, yet accurate and have high certainty factor and low confusion with less computation time. The investigation, besides having significance in soft computing research, has potential for application to large scale problems involving knowledge discovery tasks [31] and using case-based reasoning [32], particularly related to mining of classification rules.

Acknowledgments

This work was partly supported by a Council of Scientific and Industrial Research (CSIR) Grant 25(0039)/97/EMR-II, India.

References

[1] L.A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing," *Comm. ACM,* vol. 37, pp. 77-84, 1994.
[2] Z. Pawlak, *Rough Sets, Theoretical Aspects of Reasoning about Data.* Dordrecht: Kluwer Academic, 1991.
[3] Z. Pawlak, "Rough Sets," *Int'l J. Computer and Information Sciences,* vol. 11, pp. 341-356, 1982.
[4] *Rough Fuzzy Hybridization: New Trends in Decision Making.* S.K. Pal and A. Skowron, eds., Singapore: Springer Verlag, 1999.
[5] M. Banerjee, S. Mitra, and S.K. Pal, "Rough Fuzzy MLP: Knowledge Encoding and Classification," *IEEE Trans. Neural Networks,* vol. 9, no. 6, pp. 1203-1216, 1998.
[6] "Rough-Neuro Computing," *Neurocomputing,* S.K. Pal, W. Pedrycz, A. Skowron, and R. Swiniarski, eds., vol. 36, pp. 1-4, special issue, 2001.

[7] H.S. Nguyen, M. Szczuka, and D. Slezak, "Neural Network Design: Rough Set Approach to Real-Valued Data," *Proc. First European Symp. Principles of Data Mining and Knowledge Discovery (PKDD '97)*, pp. 359-366, 1997.
[8] M. Szczuka, "Refining Classifiers with Neural Networks," *Int'l J. Computer and Information Sciences*, vol. 16, pp. 39-56, 2001.
[9] L. Han, J.F. Peters, S. Ramanna, and R. Zhai, "Classifying Faults in High Voltage Power Systems: A Rough Fuzzy Neural Computational Approach," *New Directions in Rough Sets, Data Mining, and Granular Soft Computing*, pp. 47-54, 1999.
[10] J.F. Peters, A. Skowron, L. Han, and S. Ramanna, "Towards Rough Neural Computing Based on Rough Neural Networks," *Proc. Int'l Conf. Rough Sets and Current Trends in Computing (RSTC '00)*, pp. 572-579, 2000.
[11] S.K. Pal and D. Bhandari, "Selection of Optimum set of Weights in a Layered Network Using Genetic Algorithms," *Information Sciences*, vol. 80, pp. 213-234, 1994.
[12] *Genetic Algorithms for Pattern Recognition.* S.K. Pal and P.P. Wang, eds. Boca Raton, Fla.: CRC Press, 1996.
[13] A.B. Tickle, R. Andrews, M. Golea, and J. Diederich, "The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks," *IEEE Trans. Neural Networks*, vol. 9, pp. 1057-1068, 1998.
[14] S. Mitra and Y. Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework," *IEEE Trans. Neural Network*, vol. 11, pp. 748-768, 2000.
[15] B.M. Happel and J.J. Murre, "Design and Evolution of Modular Neural Network Architectures," *Neural Networks*, vol. 7, pp. 985-1004, 1994.
[16] L.M. Fu, "Knowledge-Based Connectionism for Revising Domain Theories," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, pp. 173-182, 1993.
[17] G.G. Towell and J.W. Shavlik, "Extracting Refined Rules from Knowledge-Based Neural Networks," *Machine Learning*, vol. 13, pp. 71-101, 1993.
[18] H. Liu and S.T. Tan, "X2R: A Fast Rule Generator," *Proc. IEEE Int'l Conf. System Man Cybernetics*, pp. 215-220, 1995.
[19] S.K. Pal and S. Mitra, "Multi-Layer Perceptron, Fuzzy Sets and Classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683-697, 1992.
[20] S.K. Pal and D. Dutta Majumder, *Fuzzy Mathematical Approach to Pattern Recognition.* New York: John Wiley (Halsted Press), 1986.
[21] A. Skowron and C. Rauszer, "The Discernibility Matrices and Functions in Information Systems," *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, R. Slowinski, ed., Dordrecht: Kluwer Academic, pp. 331-362, 1992.
[22] A. Skowron and J. Stepaniuk, "Decision Rules Based on Discernibility Matrices and Decision Matrices," *Proc. Int'l Workshop Rough Sets and Soft Computing*, pp. 602-609, 1994.
[23] Q. Zhao, "A Co-Evolutionary Algorithm for Neural Network-Learning," *Proc. IEEE Int'l Conf. Neural Networks*, pp. 432-437, 1997.
[24] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, Mass.: Addison-Wesley, 1989.
[25] I.A. Taha and J. Ghosh, "Symbolic Interpretation of Artificial Neural Networks," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, pp. 448-463, 1999.
[26] R. Andrews, J. Diederich, and A. B. Tickle, "A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," *Knowledge-Based Systems*, vol. 8, pp. 373-389, 1995.
[27] G.H. Rosenfeld and K. Fitzpatrick-Lins, "Coefficient of Agreement as a Measure of Thematic Classification Accuracy," *Photogrammetric Engineering and Remote SensingIntroduction*, vol. 52, pp. 223-227, 1986.
[28] E.L. Lehmann, *Testing of Statistical Hypotheses.* New York: John Wiley, 1976.
[29] A. Aspin, "Tables for Use in Comparisons Whose Accuracy Involves Two Variances," *Biometrika*, vol. 36, pp. 245-271, 1949.
[30] J. Ross Quinlan, *C4. 5, Programs for Machine Learning.* Calif.: Morgan Kaufman, 1993.
[31] S. Mitra, S.K. Pal, and P. Mitra, "Data Mining in Soft Computing Framework: A Survey," *IEEE Trans. Neural Networks*, vol. 13, pp. 3-14, 2001.
[32] *Soft Computing in Case Based Reasoning.* S.K. Pal, T.S. Dillon, and D.S. Yeung, eds., London: Springer Verlag, 2001.

Sankar K. Pal (M'81-SM"84-F'93) received the M Tech and PhD degrees in radio physics and electronics in 1974 and 1979, respectively, from the University of Calcutta. In 1982, he received another PhD degree in electrical engineering along with DIC from Imperial College, University of London. He is a professor and distinguished scientist at the Indian Statistical Institute, Calcutta. He is also the founding head of the Machine Intelligence Unit. He worked at the University of California, Berkeley and the University of Maryland, College Park during 1986-87 as a Fulbright Postdoctoral Visiting Fellow, at the NASA Johnson Space Center, Houston, Texas during 1990-1992 and 1994, as a Guest Investigator under the NRC-NASA Senior Research Associateship program. Dr. Pal is a fellow of the IEEE, the Third World Academy of Sciences, Italy, and all the four National Academies for Science/Engineering in India. His research interests includes pattern recognition, image processing, data mining, soft computing, neural nets, genetic algorithms, and fuzzy systems. He is a coauthor/coeditor of eight books including *Fuzzy Mathematical Approach*, *Pattern Recognition*, John Wiley (Halsted), New York, 1986, *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*, John Wiley, New York, 1999. He has received the 1990 S.S. Bhatnagar Prize (which is the most coveted award for a scientist in India). Dr. Pal has been an associate editor of the *IEEE Transactions on Neural Networks* (1994-1998), *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *International Journal of Pattern Recognition and Artificial Intelligence*, *Neurocomputing, Applied Intelligence, Information Sciences, Fuzzy Sets and Systems, and Fundamenta Informaticae*. He has been a member of the executive advisory editorial board of the *IEEE Transactions on Fuzzy Systems*, the *International Journal on Image and Graphics*, and the *International Journal of Approximate Reasoning*, and a guest editor of the *IEEE Computer*.

Sushmita Mitra (M'99-SM'00) received the BSc degree (Hons.) in physics and the BTech and MTech degrees in computer science from the University of Calcutta in 1984, 1987, and 1989, respectively, and the PhD degree in computer science from the Indian Statistical Institute, Calcutta in 1995. From 1992 to 1994, she was with the European Laboratory for Intelligent Techniques Engineering, Aachen, as a German Academic Exchange Service (DAAD) fellow. Since 1995, she has been an associate professor at the Indian Statistical Institute, Calcutta, where she joined in 1989. She was a recipient of the National Talent Search Scholarship (1978-83) from the National Council for Educational Research and Training, India, the IEEE TNN Outstanding Paper Award in 1994, and the CIMPA-INRIA-UNESCO Fellowship in 1996. She is a coauthor of the book *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing Paradigm* published by John Wiley, New York, in 1999, and has about fifty research publications. She was a visiting professor at Meiji University, Japan in 1999. Her research interests include data mining, pattern recognition, fuzzy sets, artificial intelligence, neural networks, and soft computing. She is a senior member of the IEEE.

Pabitra Mitra obtained the BTech degree in electrical engineering from Indian Insitute of Technology, Kharagpur in 1996. He worked as a scientist with the Center for Artificial Intelligence and Robotics, Bangalore, India. Currently, he is a Senior Research Fellow of Indian Statistical Institute, Calcutta. His research interests are in the area of data mining and knowledge discovery, pattern recognition, learning theory, and soft computing. He is a student member of the IEEE.