A Neuro-Fuzzy Scheme for Simultaneous Feature Selection and Fuzzy Rule-Based Classification

Debrup Chakraborty and Nikhil R. Pal, *Senior Member, IEEE*

***Abstract*—Most methods of classification either ignore feature analysis or do it in a separate phase, offline prior to the main classification task. This paper proposes a neuro-fuzzy scheme for designing a classifier along with feature selection. It is a four-layered feed-forward network for realizing a fuzzy rule-based classifier. The network is trained by error backpropagation in three phases. In the first phase, the network learns the important features and the classification rules. In the subsequent phases, the network is pruned to an "optimal" architecture that represents an "optimal" set of rules. Pruning is found to drastically reduce the size of the network without degrading the performance. The pruned network is further tuned to improve performance. The rules learned by the network can be easily read from the network. The system is tested on both synthetic and real data sets and found to perform quite well.**

***Index Terms*—Classification, feature analysis, neuro-fuzzy systems, rule extraction.**

I. Introduction

A classifier is a function $\mathbf{D} : R^s \rightarrow N_{pc}$ where N_{pc} is a set of label vectors defined as

$$N_{pe}\left\{\mathbf{y} \in R^c : y_i \in [0,1] \forall i \;\&\; \exists i, y_i > 0\right\} = [0,1]^c - \{0\}.$$

In other words, a classifier is a function which takes as input an object data, i.e., a feature vector in s dimension and assigns a class label to it. More specific types of label vectors can be used for specific types of classifiers as

$$N_{fc} = \left\{\mathbf{y} \in N_{pc} : \sum_{i=1}^{c} y_i = 1\right\};$$
$$N_{hc} = N_{hc}\left\{\mathbf{y} \in N_{fc} : y_i \in \{0,1\} \forall i\right\}.$$

$\mathbf{D}$ is called a crisp classifier if the classifier assigns the object to one of the classes without any ambiguity, i.e., if $\mathbf{D}[R^p] = N_{hc}$; otherwise, the classifier is fuzzy, probabilistic, or possibilistic, which together are sometimes called soft classifiers [3]. Designing a classifier means finding a good $\mathbf{D}$. $\mathbf{D}$ may be an analytical function (like the Bayes classifier) or it can be a computational transform which does classification implicitly. Fuzzy systems, neural networks, neuro-fuzzy systems or other hybrid systems are examples of such computational transforms.

Fuzzy systems built on fuzzy rules have been successfully applied to various classification tasks [3], [6], [11], [12], [34]. Fuzzy systems depend on linguistic rules which are provided by experts or the rules are extracted from a given training data set by a variety of methods like exploratory data analysis, evolutionary algorithms etc. [5], [41], [45]–[47], [51]. Fuzzy systems can be efficiently implemented by neural networks and such systems are broadly named as neural fuzzy systems [14], [15], [17], [20], [21], [29], [30], [36], [42], [43].

Owing to their specific implementation, neural networks are capable of learning input-output mappings (for classification or function approximation tasks) through minimization of a suitable error function, e.g., by means of backpropagation training algorithm. Unfortunately, neural networks are not able to learn or represent knowledge explicitly, which a fuzzy system can do through fuzzy if-then rules. Here we are making a clear distinction between identifying a mapping and extraction of readable or intelligible information or knowledge. Thus, an integration of neural networks and fuzzy systems can yield systems, which are capable of learning and decision making. In this paper we present a neural fuzzy system for classification. The various layers of the network perform different functions of a fuzzy system, also the network learns the rules required for the classification task. The rules learned by the network can be easily read from the network. Also the network has an *inherent* methodology to select the important features in the given data set, which forms an important phase for any classification process. To the best of our knowledge, no such neuro-fuzzy system exists till date.

It is known that all features that characterize a data point may not have the same impact with regard to its classification, i.e., some features may be redundant and also some may have derogatory influence on the classification task. Thus, selection of a proper subset of features from the available set of features is important for designing efficient classifiers. Methodologies for selecting good features on the basis of feature ranking etc. do exists [8], [37], [44], but most of these methods perform feature analysis in a separate phase, offline with the main classification process. Authors in [48] proposed an iterative scheme for client preference modeling. This scheme incorporates feature analysis. Here a score function is computed for each feature and in each iteration of the modeling, the feature which has the highest score from among the set of unused features is selected and used. In this method the feature selection is *not* done in an on-line manner, and in spirit the method is like feature ranking. Consequently, the set of features the model lands up with may not be the best subset of features for the task, but the process does generate a good solution to the problem. The goodness of a feature depends on the problem being solved and also on the tools being used to solve the problem [8]. Therefore, if a method

Manuscript received July 26, 2001; revised March 3, 2003.

The authors are with the Electronics and Communication Science Unit, Indian Statistical Institute, Calcutta 700108, India (e-mail: debrup_r@isical.ac.in; nikhil@isical.ac.in).

Digital Object Identifier 10.1109/TNN.2003.820557

can do feature selection simultaneously with designing the classifier, it would be able to select the most appropriate set of features for the task and can result in a good classifier. In [39] Pal and Chintalpudi developed an integrated feature selection and classification scheme based on the multilayer perceptron architecture. The present paper is motivated by this work though the formulation and philosophy of feature selection are completely different. Krishnapuram and Lee in [27] also developed a neural network for classification, which uses fuzzy aggregation functions. Their network is capable of feature selection in certain conditions though it is not primarily meant for the feature selection task. Similar networks are also discussed in [19], [25], [26], [28].

In [4] we discussed a methodology for simultaneous feature analysis and system identification in a five-layered neuro-fuzzy framework. The feature selection strategy which we use in the current work is the same as in [4]. The network in [4] was meant for system identification problems, which has been modified here to solve a classification task. Unlike the network in [4], this is a four-layered network. Also the methods used to optimize the network have been modified to suite the classification process. In this context a few *new concepts* have also been introduced for pruning of the network and, hence, the rule-base.

Our network is trained in three phases. In phase I, starting with some coarse definition of initial membership functions, the network selects important features and learns the initial rules. In phase II, the redundant nodes as detected by the feature attenuators are pruned, and the network is retuned to gain performance in its reduced architecture. In phase III, the architecture is further reduced by pruning incompatible rules, zero rules and less used rules. After pruning the network represents the final set of rules. The membership functions which constitute the final rules are now tuned to acheive better performance.

In the next section, we describe the structure of fuzzy rules for classification and also the network architecture to realize them. In Section III, we derive the rules for selecting the important features and the initial rules for classification. In Section IV, we discuss a pruning algorithm to get rid of the redundant nodes. Section V discusses two more pruning strategies along with a scheme for tuning of the membership function parameters. Section VI gives the simulation results on a synthetic data set and three real data sets. Finally, the paper is concluded in Section VII.

II. The Network Structure

Let there be s input features $(x_1, x_2, \ldots, x_s)$ and c classes $(t_1, t_2, \ldots, t_c)$. Given a $\mathbf{x} \in R^s$, the proposed neural-fuzzy system deals with fuzzy rules of the form R_i: If x_1 is A_{1i} and x_2 is $A_{2i} \ldots$ and x_s is A_{si} then $\mathbf{x}$ belongs to class t_l with a certainty d_l, $(1 \leq l \leq c)$. Here, A_{ji} is the ith fuzzy set defined on the domain of x_j.

From our notation one might think that for each rule we are using a different set of antecedent linguistic values (fuzzy sets) but that is not necessarily true; in fact, for every feature only a few fuzzy sets are defined and, hence, some of the $A_{ji} = A_{jh}$ for some i and h.

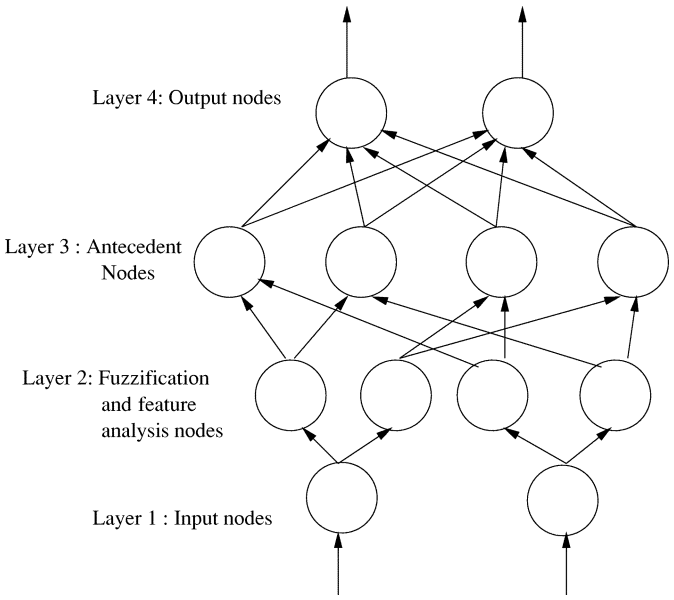

Fig. 1. The network structure.

Note that, the structure of the rules used here is quite different from the structure of the rules used in [4]. The proposed neural-fuzzy system is realized using a four layered network as shown in Fig. 1. The first layer is the input layer, the second layer is the membership function and feature selection layer, the third layer is called the antecedent layer and the fourth layer is the output layer. The activation function of each node with its inputs and outputs, is discussed next layer by layer. We use suffixes p, n, m, l to denote, respectively, the suffixes of the nodes in layers 1 through 4. The output of each node is denoted by z.

Layer 1: Each node in layer 1 represents an input linguistic variable of the network and is used as a buffer to transmit the input to the next layer, that is to the membership function nodes of its linguistic values. Thus, the number of nodes in this layer is equal to s. Let x_p be the input to the p^{th} node in layer 1 then the output of the node will be

$$z_p = x_p. \tag{1}$$

Layer 2: This is the fuzzification and feature analysis layer, which is similar to the layer 2 of the network described in [4]. Each node in this layer represents the membership function of a linguistic value associated with an input linguistic variable. The output of a layer 2 node represents the membership grade of the input with respect to a linguistic value. We use bell shaped membership functions. All connection weights between the nodes in layer 1 and layer 2 are unity. If there be N_i fuzzy sets associated with the i^{th} feature then the number of nodes in this layer is $\sum_{i=1}^{s} N_i$. The output of a node in layer 2 is computed by

$$\bar{z}_n \exp\left\{-\frac{(z_p - \mu_n)^2}{\sigma_n^2}\right\}. \tag{2}$$

Here μ_n and σ_n are the center and spread, respectively, of the bell shaped function representing a term of the linguistic variable x_p associated to node n (n indicates the nth term (fuzzy set) of the linguistic variable x_p).

As in [4], for feature selection, the output of this layer needs to be modified. The activation function of any node n in layer 2 is taken as

$$z_n = \bar{z}_n^{\left(1-e^{-\beta_p^2}\right)}. \tag{3}$$

The justification behind (3) can be found in [4]. The tunable parameter β_p associated with each input feature x_p is called a feature modulator. β_ps are learned by backpropagation. When β_p^2 takes a large value then z_n tends to $\bar{z}_n$ and for small values of β_p^2, z_n tends to 1, thereby making the feature indifferent. Therefore, our objective would be to make β_p^2 take large values for good features and small values for bad ones through the process of learning. Up to this the functioning of the present network is similar to that in [4].

In learning phase I, the parameters of the membership functions are kept fixed and only the β_ps are learned through error backpropagation. In learning phase III, the β_ps are kept fixed and the membership function parameters are updated. Initially we define the parameters of the membership functions in an *ad hoc* manner. We choose an arbitrary number of fuzzy sets, with considerable overlap between adjacent ones such that the antecedents span the entire range of variability (bounding hyperbox) of the input features. This conservative initialization strategy may require further optimization of the membership function parameters, which is done in learning phase III.

Layer 3: This layer is called the antecedent layer. Each node in this layer represents the IF part of a fuzzy rule. The number of nodes in this layer is $\prod_{i=1}^{s} N_i$. There are many operators $(T-norms)$ for fuzzy intersection [22], [32] of which min and product are quite popular and have been widely used in numerous applications. In [4] we used product as the operator for intersection and got fairly good results. But we know that for any $T-norm, \mathcal{T}, \mathcal{T}(x,y) \leq min(x,y)$. So use of product as the intersection operator is counter-intuitive. To elaborate it, consider two propositions with truth values a and b. It is not natural to assume that they will produce a firing strength less than $min(a,b)$. Let us consider a rule:

If x_1 is A_1 and x_2 is A_2 *and* $\ldots x_s$ is A_s then the class is K.

Suppose, for an input $\mathbf{x} = (x_1, x_2, \ldots, x_s)^T$ each x_i, $i = 1, 2, \ldots, s$, has a membership of 0.9 in the respective fuzzy set $A_i, i = 1, 2, \ldots, s$. Thus, for this input if *product* is used as the operator for intersection then the firing strength of the rule will be $(0.9)^s$. So, the firing strength decreases exponentially with s. So, for a reasonably big s, the firing strength reduces almost to zero, though each of the input components has a high membership of 0.9 in the corresponding fuzzy set. Therefore, the use of product for intersection is not intuitively appealing. One might wonder why did we (others also) get good results using product in [4]. The answer lies in the defuzzification process. For example, in the height method of defuzzification, the defuzzified value is computed as a weighted sum of the peaks of the output fuzzy sets, where the weights are the normalized values of the firing strengths. The final output is, thus, computed as a convex combination of the peaks of the output fuzzy sets. But in this case such defuzzification methods cannot be applied. Consequently, here we use min as the operator for intersection. As min is not differentiable, for ease of computation many softer versions of min that are differentiable have been previously used [2], [38]. We use the following soft version of min which we call $softmin$:

$$\text{softmin}(x_1, x_2, \ldots, x_s, q) = \left(\frac{x_1^q + x_2^q + \ldots + x_s^q}{s}\right)^{\frac{1}{q}}.$$

As $q \to -\infty$, $softmin$ tends to the minimum of all x_is, $i = 1, 2, \ldots, s$. For our purpose we use $q = -12$ in all results reported. Note that, $softmin$ is not a $T-norm$ as it does not satisfy the criteria of associativity and identity. For our feature selection task, the intersection operator must satisfy the property of $identity$, which $softmin$ does possess for $q \to -\infty$. So, $softmin$ is compatible with our feature selection strategy though it is not a $T-norm$. Thus, the output of the mth node in layer 3 is

$$z_m = \left(\frac{\sum_{n \in P_m} z_q^n}{|P_m|}\right)^{\frac{1}{q}} \tag{4}$$

where P_m is the set of indexes of the nodes in layer 2 connected to node m of layer 3 and $|P_m|$ denotes the cardinality of P_m.

Layer 4: This is the output layer and each node in this layer represents a class. Thus, if there are c classes then there will be c nodes in layer 4. The nodes in this layer perform an OR operation, which combine the antecedents of layer 3 with the consequents. According to the structure of the fuzzy rules that we are concerned about, the consequent of a rule is a class with a degree of certainty. Thus, the output of node l in this layer represents the certainty with which a data point belongs to class l. We classify a point $\mathbf{x}$ to a class q, if $z_q = max_l(z_l)$. The nodes in layers 3 and 4 are fully connected. Let w_{lm} be the connection weight between node m of layer 3 and node l of layer 4. The weight w_{lm} represents the certainty factor of a fuzzy rule, which comprises the antecedent node m in layer 3 as the IF part and the output node l in layer 4 representing the THEN part. These weights are adjustable while learning the fuzzy rules. The OR operation is performed by some $\mathrm{s-norm}$ [22]. We use here the max operator. Thus the output of node l in layer 4 is computed by

$$z_l = max_{m \in P_l}(z_m w_{lm}) \tag{5}$$

where P_l represents the set of indexes of the nodes in layer 3 connected to the node l of layer 4. Since w_{lm}s are interpreted as certainty factors, each w_{lm} should be nonnegative and it should lie in [0, 1]. The error backpropagation algorithm or any other gradient based search algorithm does not guarantee that w_{lm} will remain nonnegative, even if we start the training with nonnegative weights. Hence, we model w_{lm} by $e^{-g_{lm}^2}$. The g_{lm} is unrestricted in sign but the effective weight $w_{lm} = e^{-g_{lm}^2}$ will always be nonnegative and lie in [0, 1]. Therefore, the output (activation function) of the lth node in layer 4 will be

$$z_l = max_{m \in P_l}\left(z_m e^{-g_{lm}^2}\right). \tag{6}$$

Since it is enough to pick the node with the maximum value of the product of firing strength and certainty factor, it is not necessary to maintain w_{lm} in [0, 1]. The nonnegativity alone would be enough. So we use $w_{lm} = g_{lm}^2$.This also reduces

the computational overhead on the network. Thus, (6) can be modified to

$$z_1 = max_{m \in P_l} \left(z_m g_{lm}^2 \right). \tag{7}$$

Note that, in layer 4 we use the usual max operator instead of a differentiable soft version of max. As max is not differentiable the update equations that we derive in the next section have to be split for different conditions of the network. We could have used a differentiable (soft) version of max here too. But if we use both soft version of min and soft version of max as activation functions in two layers of the network, the update equations would become very complicated. Hence, we choose to use the usual max in this layer and a differentiable version of min in layer 3. The learnable weights w_{lm} are updated in all the three learning phases.

III. Learning Phase I: Feature Selection and Rule Detection

We now derive the learning rules for the neural-fuzzy classifier with the activation or node functions described in the previous section. All the training phases use the concept of backpropagation to minimize the error function

$$e = \frac{1}{2} \sum_{i=1}^{N} E_i = \frac{1}{2} \sum_{i=1}^{N} \sum_{l=1}^{c} (y_{il} - z_{il})^2 \tag{8}$$

where c is the number of nodes in layer 4 and y_{il} and z_{il} are the target and actual outputs of node l in layer 4 for input data $\mathbf{x}_i$; $i = 1, 2, \ldots, N$. In learning phase I the the learnable weights between layer 3 and layer 4 and the parameters β_ps in layer 2 are updated based on gradient descent search. We use online update scheme and, hence, derive the learning rules using the instantaneous error function E_i. Without loss of generality we drop the subscript i in our subsequent discussions.

The delta value δ of a node in the network is defined as the influence of the node output on E. The derivation of the delta values and the adjustment of the weights and β_ps are presented layer wise next.

Layer 4: The output of the nodes in this layer is given by (7) and the δ value for this layer δ_l will be

$$\delta_1 = \frac{\partial E}{\partial z_l}.$$

Thus,

$$\delta_l = -(y_l - z_l). \tag{9}$$

Layer 3: The delta for this layer is

$$\delta_m = \frac{\partial E}{\partial z_m} = \frac{\partial E}{\partial z_l} \frac{\partial z_l}{\partial z_m}.$$

Hence, the value of δ_m will be

$$\delta_m = \begin{cases} \sum_{l \in Q_m} \delta_l g_{lm}^2, & \text{if } z_m g_{lm}^2 = \max_{m'} \left\{ z_{m'} g_{lm'}^2 \right\} \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Here Q_m is the set of indexes of the nodes in layer 4 connected with node m of layer 3.

Layer 2: Similarly, the δ_n for layer 2 is

$$\delta_n = \frac{\partial E}{\partial z_n} = \frac{\partial E}{\partial z_m} \frac{\partial z_m}{\partial z_{zn}}.$$

Hence,

$$\delta_n = \sum_{m \in R_n} \delta_m \left(\frac{z_m z_n^{q-1}}{\sum_{n \in P_m} z_n^q} \right). \tag{11}$$

In (11) R_n is the set of indexes of nodes in layer 3 connected with node n in layer 2.

With the δ calculated for each layer now we can write the weight update equation and the equation for updating β_p.

$$\frac{\partial E}{\partial g_{lm}} = \frac{\partial E}{\partial z_l} \frac{\partial z_l}{\partial g_{lm}},$$

or

$$\frac{\partial E}{\partial g_{lm}} = \begin{cases} \sum_{l \in Q_m} 2 \delta_l z_m g_{lm}, & \text{if } z_m g_{lm}^2 = \max_{m'} \left\{ z_{m'} g_{lm'}^2 \right\} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Similarly, we calculate

$$\frac{\partial E}{\partial \beta_p} = \frac{\partial E}{\partial z_n} \frac{\partial z_n}{\partial \beta_p},$$

or

$$\frac{\partial E}{\partial \beta_p} = - \sum_{n \in R_p} \delta_n \left(2 \beta_p e^{-\beta_p^2 z_n} \right) \left(\frac{z_p - \mu_n}{\sigma_n} \right)^2. \tag{13}$$

Here, R_p is the set of indexes of nodes in layer 2 connected to node p of layer 1. Hence, the update equations for weights g_{lm} and β_p are

$$g_{lm}(t+1) = g_{lm}(t) - \eta \left(\frac{\partial E}{\partial g_{lm}} \right) \tag{14}$$

and

$$\beta_p(t+1) = \beta_p(t) - \nu \left(\frac{\partial E}{\partial \beta_p} \right). \tag{15}$$

In (14) and (15), η and ν are learning coefficients, which are usually chosen by trial and error or one can use methods described in [10], [30] for better choices.

The network learns the weights of the links connecting layers 3 and 4 and also the parameters associated with nodes in layer 2, which do the feature selection. The initial values of βs can be so selected that no feature gets into the network in the beginning and the learning algorithm will pass the features, which are important, i.e., the features that can reduce the error rapidly.

A. Implicit Tuning of Membership Functions

The feature modulators not only help us to select good features but also have an interesting side effect. They tune the membership functions to some extent. The output of a layer 2 node is

$$z_n = (\bar{z}_n)^{\gamma_p}$$

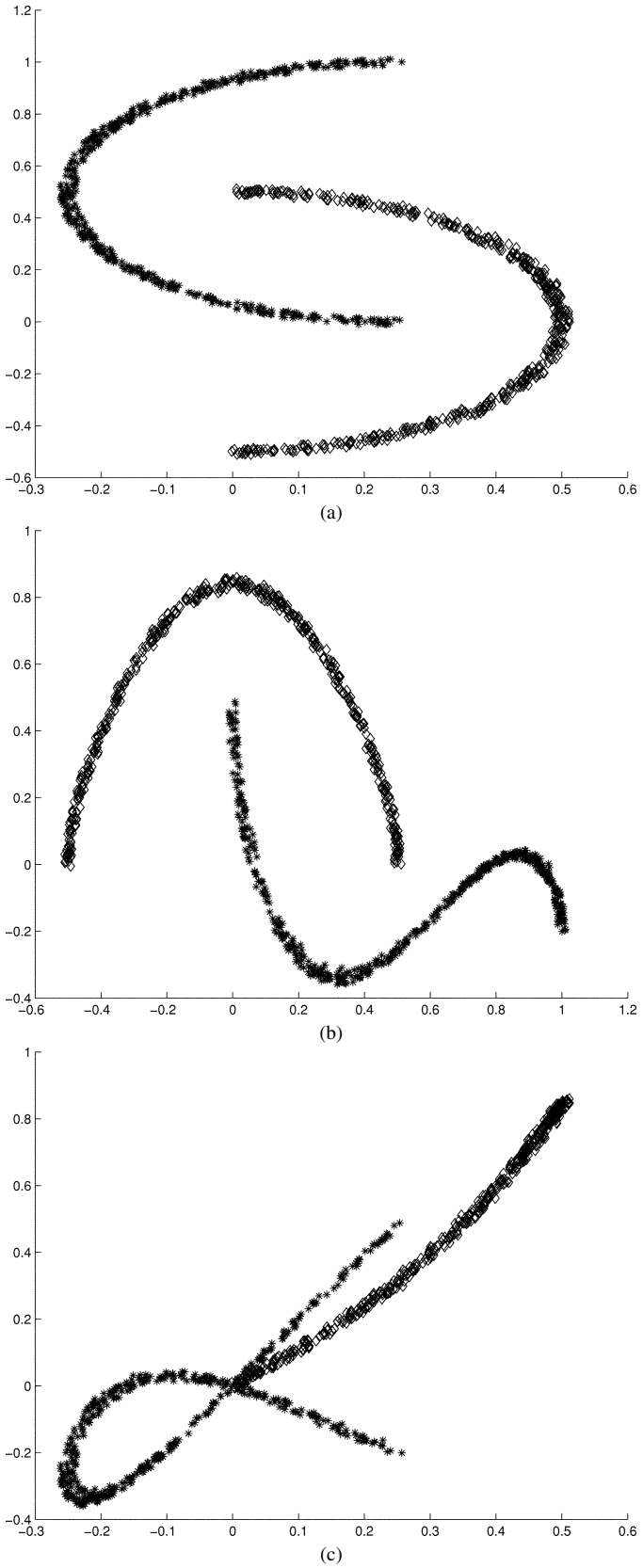

Fig. 2. Plot of ELONGATED. (a) Features 1–2 . (b) Features 2–3. (c) Features 1–3.

where

$$\bar{z}_n = exp\left\{-\frac{(z_p-\mu_n)^2}{\sigma_n^2}\right\}$$

and $\gamma_p = 1 - e^{-\beta_p^2}$.

Note that, z_n is also a Gaussian function with mean μ_n and spread σ_n' because

$$\left[exp\left\{-\frac{(x-\mu_n)^2}{{\sigma_n}^2}\right\}\right]^{\gamma_p} = exp\left\{-\frac{(x-\mu_n)^2}{{\sigma_n'}^2}\right\} \tag{16}$$

where

$$\sigma_n' = \frac{\sigma_n}{\sqrt{\gamma_p}}. \tag{17}$$

The feature modulators, thus, tune the spread of the input membership functions and retain their Gaussian nature.

At this point a natural question may come: why are we not tuning the parameters of the membership functions at this phase of training? There are two reasons. First, tuning of spreads of the membership functions although can reduce the misclassifications, it cannot do the task of feature selection. Tuning of membership function parameters refines each rule to reduce the classification error, while for elimination of a feature the spreads of all membership functions defined on that feature need to be modified. Second, simultaneous tuning of modulator functions and the parameters like mean and spread of membership functions is not desirable. Because, tuning of modulator function looks at all membership functions defined on a feature as a whole, while the tuning of membership function parameters tries to improve the performance of a rule, i.e., tunes parameters of the membership functions considering each rule separately. As a result, if both modulator functions and membership function parameters are tuned, the learning process may become unstable. Therefore, tuning of parameters of membership functions should be done after feature elimination (i.e., after tuning of the modulator functions). Consequently, tuning of membership function parameters are done in learning phase III which is discussed later.

Next we discuss strategies to prune the network to make it more readable.

IV. Leaning Phase II: Pruning of Redundant Nodes and Further Training

We started with a network which represented all possible rules. But all possible rules usually are never needed to represent a system. Moreover, the modulator functions associated with the second layer nodes, may decide that all features present are not really important. Hence, some of the nodes present in the network may be redundant. The presence of these redundant nodes will decrease the readability/interpretability of the network and add to its computational overhead. Also as per our formulation each node in layer 3 is connected with all nodes in layer 4, and this gives rise to *incompatible rules* that need to be removed. Further, here we used fuzzy sets which covered the total domain of each feature, thus the antecedents cover the entire hyperbox that bounds the data. The training data may (usually will) not be distributed even over the smallest hyperbox containing the data. Consequently, there may be some rules which are never fired by the training data. Such rules, which are not supported by the training data could be harmful. The certainty factors of such rules can be erratic and they can lead to bad generalization. So, it is necessary to get rid of *redundant nodes*, *incompatible rules* and *less used rules*.

TABLE I
SUMMARY OF THE DATA SETS

Name	Total Size	Trng. Size	Test Size	No. of classes	No. of features
ELONGATED	1000	500	500	2	3
IRIS	150	100	50	3	4
PHONEME	5404	500	4904	2	5
RS-DATA	262144	1600	260544	8	7

TABLE II
FREE PARAMETERS OF THE NETWORK

Parameter	Meaning	Initial Value	Updating phase and update equations
β_p	Modulator function for feature p	0.001	Phase I eq. (15)
g_{lm}	Certainty factor of a rule represented by antecedent node m and output node l	1.0	Phases I,II and III eq. (14)
μ_n	Center of the membership function denoted by node n of layer 2	As discussed	Phase III eq. (18)
σ_n	Spread of the membership function denoted by node n of layer 2	As discussed	Phase III eq. (19)

A. Pruning Redundant Nodes

Let us consider a classification problem with s input features. So layer 1 of the network will have s nodes. Let the indexes of these nodes be denoted by $i(i = 1\ to\ s)$. Let $\mathcal{N}_i$ be the set of indexes of the nodes in layer 2, which represents the fuzzy sets on the feature represented by node i of layer 1. We also assume that $r(r < s)$ of the s features are indifferent/bad as dictated by the training. Let R be the set of r indexes of the nodes, representing the r indifferent/bad features. Hence, any node with index i in layer 1 such that $i \in R$ is redundant. Also any node j in layer 2, where $j \in \mathcal{N}_i$ and $i \in R$ is redundant. In [4] we provided a scheme to prune redundant nodes in layer 1 and layer 2. The same scheme is applicable here. After pruning of the redundant nodes a few epochs of training is required in the reduced architecture for the network to regain its performance.

The selection of a redundant feature p depends on the value of $\gamma_p = 1 - e^{-{\beta_p}^2}$. A feature p is considered redundant and consequently a member of the set R, if $\gamma_p < th$, where th is a threshold. It was shown in [4] that $th = 0.05$ is a reasonable value for the threshold. Please refer to [4] for the details.

After pruning the redundant nodes, the certainty factor of the rules are further tuned using (14). Phase II training ends, once the values of w_{lm} stabilize.

V. LEARNING PHASE III: PRUNING OF INCOMPATIBLE RULES, LESS USED RULES, AND ZERO RULES AND FURTHER TRAINING

In phase III, the network is pruned further and the certainty factors of the rules are again tuned. In this last phase of training the parameters of the membership functions are also tuned. The details are presented in the following subsections.

A. Pruning Incompatible Rules

As per construction of our network, the nodes in layer 3 and layer 4 are fully connected and each link corresponds to a rule. The weight associated with each link is treated as the certainty factor of the corresponding rule. If there are c classes then layer 4 will have c nodes and there will be c rules with the same antecedent but different consequents, which are inherently inconsistent.

Suppose layer 3 has N^3 nodes (i.e., N^3 antecedents) and layer 4 has N^4 nodes (thus, N^4 consequents or classes). Each node m in layer 3 is then connected to N^4 nodes in layer 4. The link connecting node m of layer 3 and node l of layer 4 has a weight w_{ml} associated with it, which is interpreted as the certainty factor of the rule represented by the link. For each node m in layer 3 we retain only one link with a node in layer 4 that has the highest certainty factor associated with it.

B. Pruning Zero Rules and Less Used Rules

After removal of the incompatible rules each node in layer 3 is connected with only one node in layer 4. Suppose node m in layer 3, which is connected to a node l in layer 4, has a very low weight $w_{lm} < \omega_{low}$ (we take $\omega_{low} = 0.001$). Then, the rule associated with the node pair m and l has a very low certainty factor and it does not contribute significantly in the classification process. We call such rules as *Zero rules*. These rules can be removed from the network. In fact such rules should

TABLE III
USER DEFINED PARAMETERS

Parameter	Meaning	Suggested Value(s)
η	Learning constant for certainty factors of rules	Decided according to data
ν	Learning constant for feature modulators	Decided according to data
λ	Learning constant for membership function parameters	Decided according to data
τ	Threshold on the number of data points for choosing *less used* rules	3
th	Threshold for choosing redundant nodes	0.05
ω_{low}	Threshold for choosing *zero* rules	0.001

be removed from the network as we do not like to make any decision with a very low confidence. Removal of a zero rule means removing a node in layer 3 along with its links.

Further, as discussed earlier, our network starts with all possible antecedents which cover a hyperbox bounding the data. Hence, there may (usually will) be rules which are never fired or fired by only a few data points. Such rules are not well supported by the training data and will result in bad generalization. We call such rules as *less used rules* and they should also be removed. As per the structure of our network, the class label of a point is decided by only one rule. Thus, there may be rules present in the network which never takes part in decision making or makes decision for only a very few points. These rules represent parts of the input space which are not supported by training data or they represent outliers. These rules are bad rules and can lead to bad generalization. To remove these rules, for each antecedent node m (note that at this stage, an antecedent represents an unique rule) we count the number of data points n_t, for which m makes the decision. If n_t is less than τ then we consider node m as well as the rule represented by it as inadequately supported by the training data. Every such node m in layer 3 along with its links can be removed. The choice of τ is related to the definition of outliers. In this study unless a rule represents at least three points, we take it as a less-used rule and delete it.

C. Tuning Parameters of the Reduced Rule Base

The network (rule base) obtained after pruning of the redundant nodes, less used rules and the zero rules is considerably smaller than the initial network with which we began. The parameters of the membership functions of this reduced rule base are now tuned to get a better classifier performance. The update equations for the centers and the spreads of the membership functions can be written as

$$\mu_n(t+1) = \mu_n(t) - \lambda \frac{\partial E}{\partial \mu_n(t)} \tag{18}$$

TABLE IV
NUMBER OF FUZZY SETS FOR EACH FEATURE FOR ELONGATED

Feature 1	3
Feature 2	4
Feature 3	3

and

$$\sigma_n(t+1) = \sigma_n(t) - \lambda \frac{\partial E}{\partial \sigma_n(t)}. \tag{19}$$

Here

$$\frac{\partial E}{\partial \mu_n} = 2\delta_{n\gamma_p} z_n \frac{(z_p - \mu_n)}{\sigma_n^2} \tag{20}$$

and

$$\frac{\partial E}{\partial \sigma_n} = 2\delta_{n\gamma_p} z_n \frac{(z_p - \mu_n)^2}{\sigma_n^3} \tag{21}$$

where z_p is the feature related to node n of the second layer and λ is a predefined learning constant.

The update equations (18) and (19) are applied iteratively along with (14) till there is no further decrement in the error defined by (8). We emphasize that in this phase of training the β values are not updated. As discussed earlier, tuning of the feature modulators along with the membership function parameters may make the training unstable.

VI. SIMULATION RESULTS

A. The Data Sets

The methodology is tested on four data sets and the results obtained on them are quite satisfactory. Of the four data sets, one is a synthetic data set named ELONGATED and other three are real data sets named IRIS, PHONEME, and RS-DATA.

ELONGATED [31] has three features and two classes, the scatterplots of features 1–2, 2–3, and 1–3 are shown in Fig. 2.

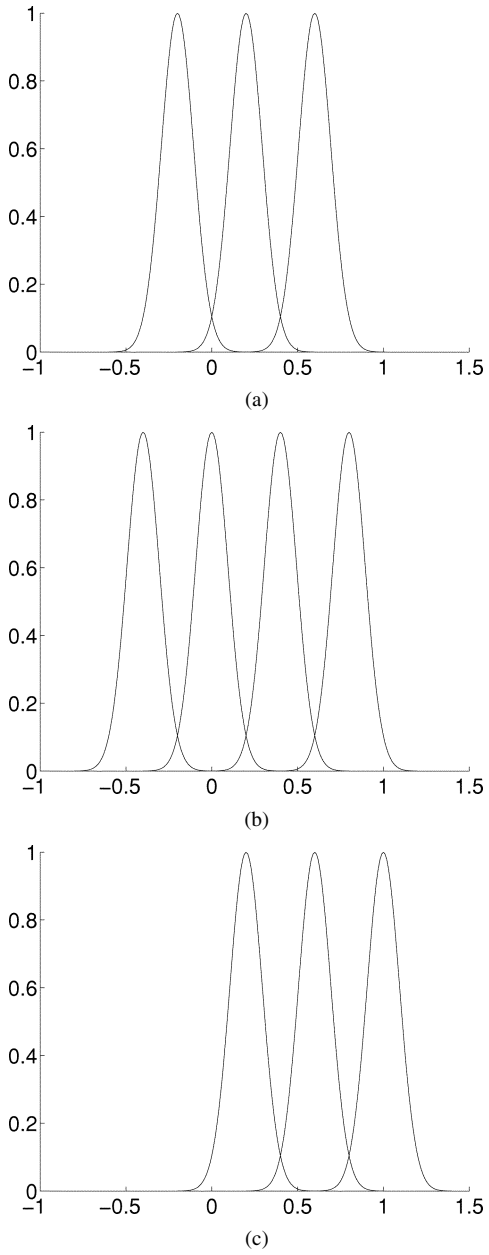

Fig. 3. Fuzzy sets used for ELONGATED. (a) Feature 1. (b) Feature 2. (c) Feature 3.

These plots show that features 1 and 2 or features 2 and 3 are enough for the classification task. Thus, any one of these two combinations is adequate to solve the problem. IRIS [1] is a data set with four features and three classes. It is well known that for IRIS features 3 and 4 are enough for the classification task [3]. The PHONEME data set contains vowels coming from 1809 isolated syllables (for example: pa, ta, pan,...) in French and Spanish language [24], [52]. Five different attributes are chosen to characterize each vowel. These attributes are the amplitudes of the five first harmonics AHi, normalized by the total energy Ene (integrated on all frequencies), AHi/Ene. Each harmonic is signed positive when it corresponds to a local maximum of the spectrum and negative otherwise. The Phoneme data set has two classes, nasal and oral.

The RS-DATA [23] is a satellite image of size 512 $\times$ 512 pixels captured by seven sensors operating in different spectral bands from Landsat-TM3. Each of the sensors generates an image with pixel values varying from 0 to 255. The 512 $\times$ 512 ground truth data provide the actual distribution of classes of objects captured in the image. From these images we produce the labeled data set with each pixel represented by a 7-D feature vector and a class label. Each dimension of a feature vector comes from one channel and the class label comes from the ground truth data.

TABLE V
INITIAL ARCHITECTURE OF THE NETWORK USED TO CLASSIFY ELONGATED

Layer 1	3
Layer 2	10
Layer 3	36
Layer 4	2

TABLE VI
VALUE OF $1 - e^{-\beta_p{}^2}$ FOR DIFFERENT INPUT FEATURES FOR ELONGATED

Features	1	2	3
$1 - e^{-\beta_p^2}$	0.51	0.44	0.01

We divide each data set X into training (X_{tr}) and test (X_{te}) sets, such that $X_{tr} \cup X_{te} = X$ and $X_{tr} \cap X_{te} = \phi$. For ELONGATED, IRIS and PHONEME the training and test divisions were made randomly. For RS-DATA we created a training sample containing exactly 200 points from each class. The summary of the data sets used is given in Table I.

B. The Implementation Details

Before we present the results we discuss a few implementational issues. The network contains the feature modulators (β_ps), the certainty factor of the rules (g_{lm}s) and the membership function parameters (μ_n and σ_n) as free parameters. The initial values of β_ps are set to 0.001; so, initially the network considers all features to be equally *unimportant*. The initial values of the certainty factor of the rules, i.e., the weights of the links between layer 3 and layer 4 are all set to 1.0, which signifies that initially all rules have the *same* certainty factor. For the initial values of the membership function parameters, for each feature we choose an arbitrary number of equidistant fuzzy sets with considerable overlap between adjacent ones. These fuzzy sets span the entire domain of the feature. The choice of fuzzy sets may have considerable influence on the performance of the classifier. For one data set with complex class structures we use exploratory data analysis to get the initial network structure. It is possible to design more elaborate methods using clustering and cluster validity indexes to decide the initial membership functions. For some such methods, readers can refer to [35], [40]. Since the thrust of this paper is to establish the utility of integrated feature selection and classifier design in a neuro-fuzzy framework, we do not pursue the issue of network initialization further. Table II depicts the list of the free parameters with their meanings and initial values.

Other than the free parameters the network also contains some user defined parameters. The learning parameters η and

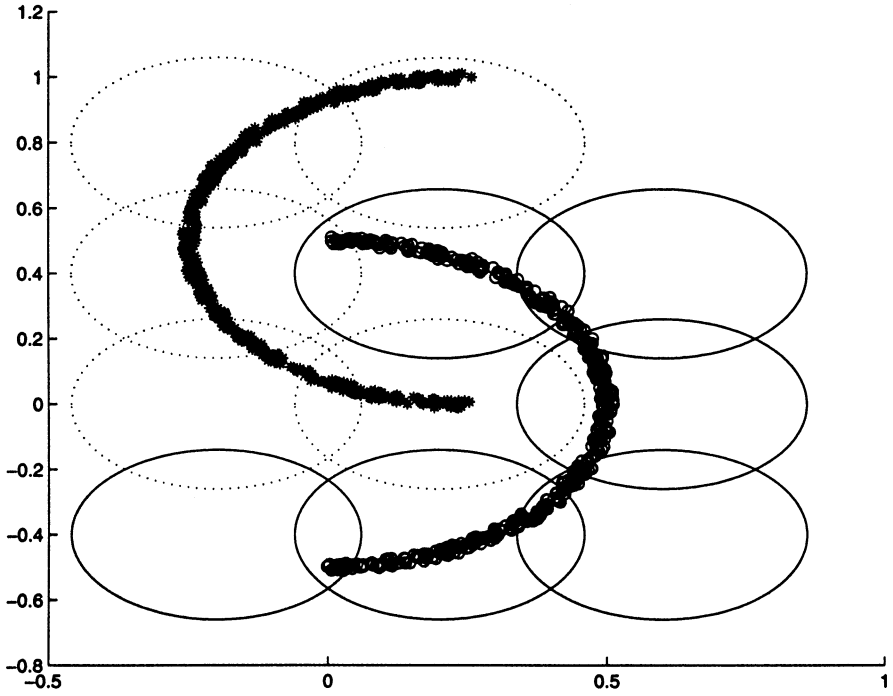

Fig. 4. The rules for classifying ELONGATED.

TABLE VII
THE LINGUISTIC RULES FOR ELONGATED

Rule no.	Rule
1	if x_1 is CLOSE TO -0.2 and x_2 is CLOSE TO -0.4 then class 1
2	if x_1 is CLOSE TO -0.2 and x_2 is CLOSE TO 0.0 then class 2
3	if x_1 is CLOSE TO -0.2 and x_2 is CLOSE TO 0.4 then class 2
4	if x_1 is CLOSE TO -0.2 and x_2 is CLOSE TO 0.8 then class 2
5	if x_1 is CLOSE TO 0.2 and x_2 is CLOSE TO -0.4 then class 1
6	if x_1 is CLOSE TO 0.2 and x_2 is CLOSE TO 0.0 then class 2
7	if x_1 is CLOSE TO 0.2 and x_2 is CLOSE TO 0.4 then class 1
8	if x_1 is CLOSE TO 0.2 and x_2 is CLOSE TO 0.8 then class 2
9	if x_1 is CLOSE TO 0.6 and x_2 is CLOSE TO -0.4 then class 1
10	if x_1 is CLOSE TO 0.2 and x_2 is CLOSE TO 0.0 then class 1
11	if x_1 is CLOSE TO 0.2 and x_2 is CLOSE TO 0.4 then class 1

ν and λ are differently chosen for different data sets. For all the data sets, as explained earlier, τ is taken as 3, th as 0.05, and ω_{low} as 0.001. Table III gives the values of all user defined parameters of the network. Other than the learning coefficients, users can use the values suggested in Table III. For the learning coefficient, as done in all gradient based learning schemes, either one can use a trial and error method or schemes discussed in [10], [13], [30], [49], [50]. A simple workable solution is to keep a snap-shot of all learnable parameters before an epoch starts and use a high value (say 1) of the learning coefficient. If the average error after an epoch is found to increase, then the learning constant is decreased by an amount, say 10% of the present value and the learning is continued after resetting the learnable parameters using the snap-shot.

C. Results

1) Results on ELONGATED: The number of fuzzy sets used for each feature for this problem is shown in Table IV and the actual fuzzy sets used are depicted in Fig. 3. The initial architecture is shown in Table V.

After only 100 iterations, the number of misclassifications on the training set were reduced to 0. The values of $1 - e^{-\beta_p{}^2}$ after 100 iterations are depicted in Table VI. Table VI shows that the network selects features 1 and 2 and rejects the third feature. Consequently the network is pruned for redundant nodes. After pruning the network retains its performance, i.e., produces a misclassification of 0 on X_{tr}.

Initially, the network had 36 antecedent nodes. Hence, it had $36 \times 2 = 72$ rules. After pruning of the redundant

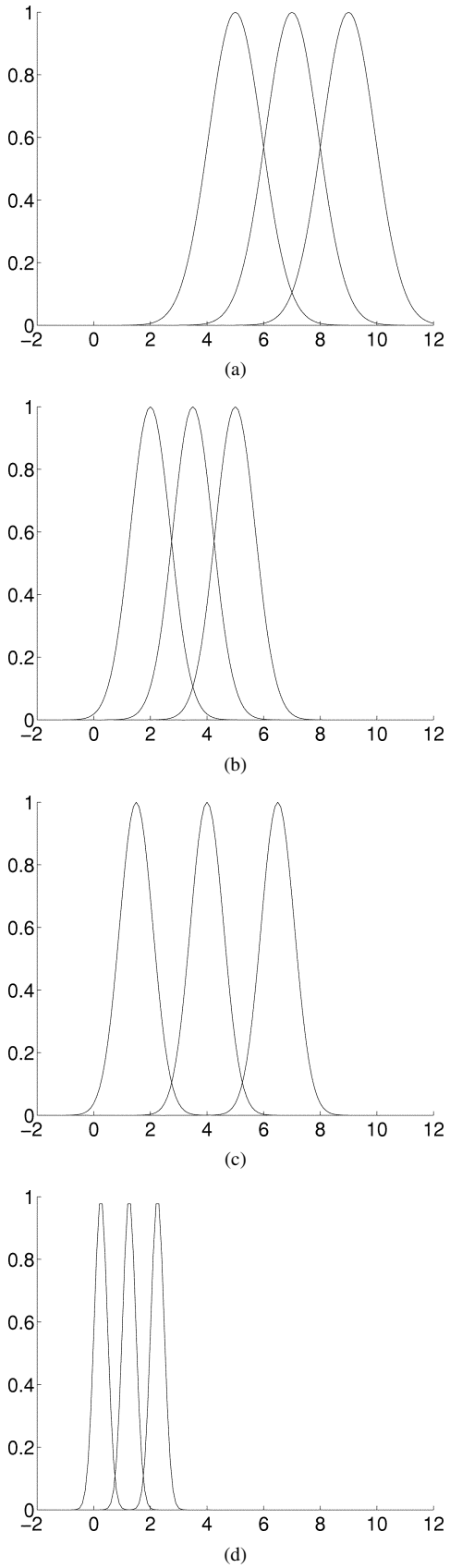

Fig. 5. Fuzzy sets used for IRIS. (a) Feature 1. (b) Feature 2. (c) Feature 3. (d) Feature 4.

nodes the number of antecedent nodes becomes 12; hence, at this stage the number of rules is $12 \times 2 = 24$. Next, the incompatible rules are pruned to obtain 12 rules. Finally, one rule is found to be *less used* and hence removed. Thus, the final architecture represents 11 rules as shown in Fig. 4. In Fig. 4 the 11 ellipses represent the 11 antecedent clauses. The coordinates of the center of an ellipse correspond to the centers of the two bell shaped fuzzy sets that form the antecedent clause. The major and minor axes of the ellipses are equal to twice the spreads of the respective fuzzy sets. For antecedent clauses of rules representing a particular class we use a particular type of line to draw the corresponding ellipses. For example, in Fig. 4, the continuous line is used to represent class 1 and the dotted line for class 2. The linguistic rules read from the network are shown in Table VII.

The final network produces a misclassification of 0 on X_{tr} as well as on X_{te}.

2) Results on IRIS: Here we used three fuzzy sets for each of the four features. The fuzzy sets are shown in Fig. 5. The initial architecture for the network is shown in Table VIII.

After 300 epochs, the β_p values stabilized and the number of misclassifications produced on the training set X_{tr} was 3. The values of $1 - e^{-\beta_p{}^2}$ after 300 iterations are shown in Table X, which suggests that only features 3 and 4 are important. Hence, we prune the redundant nodes. After pruning, the network *still* produces three misclassifications on X_{tr}.

In this case the initial network had 81 antecedent nodes resulting in $81 \times 3 = 243$ rules. After pruning of the redundant nodes the number of antecedent nodes becomes 9, hence, at this stage the number of rules is $9 \times 3 = 27$. Next, the incompatible rules are pruned to obtain nine rules. For IRIS we found four *less used* rules and we removed them. The final architecture represented only five rules that are depicted in Fig. 6 and the corresponding linguistic rules are shown in Table XI. In Iris data features 3 and 4 represent petal length and petal width of iris flowers, respectively. In Table XI pl and pw represent the petal length and petal width, respectively. The final network again produces a misclassification of 3 on the training set X_{tr} and 1 on the test set X_{te}. This clearly suggests that pruning neither degrades the performance on the training data nor the generalization capability of the system.

We also trained our network with all the 150 points. In this case too we obtained just the same results in terms of features selected and misclassification, i.e., we obtained a misclassification of 4 on the entire data set.

There are many results available on Iris data in the literature. Table IX (adopted from [43])shows the best resubstitution accuracy for some rule based classifiers. From Table IX it is clear that FuGeNeSys and SuPFuNIS show the best results using five rules. We obtain a resubstitution accuracy of 97.3% with five rules which is better than all others in Table IX leaving out FuGeNeSys and SuPFuNIS. Note that, all results in Table IX are obtained by using all four features, but our classifier uses only two features. In [7], Chiu uses only two features, i.e., features 3 and 4 to develop a classifier. The features considered by Chiu are the same as those selected by our classifier. Chiu reports a training error of 3 and a test error of 0 on a training test partition of $120 + 30$ using only three rules. This is of course better than the result obtained by us, but we do not know the training test partition used in [7].

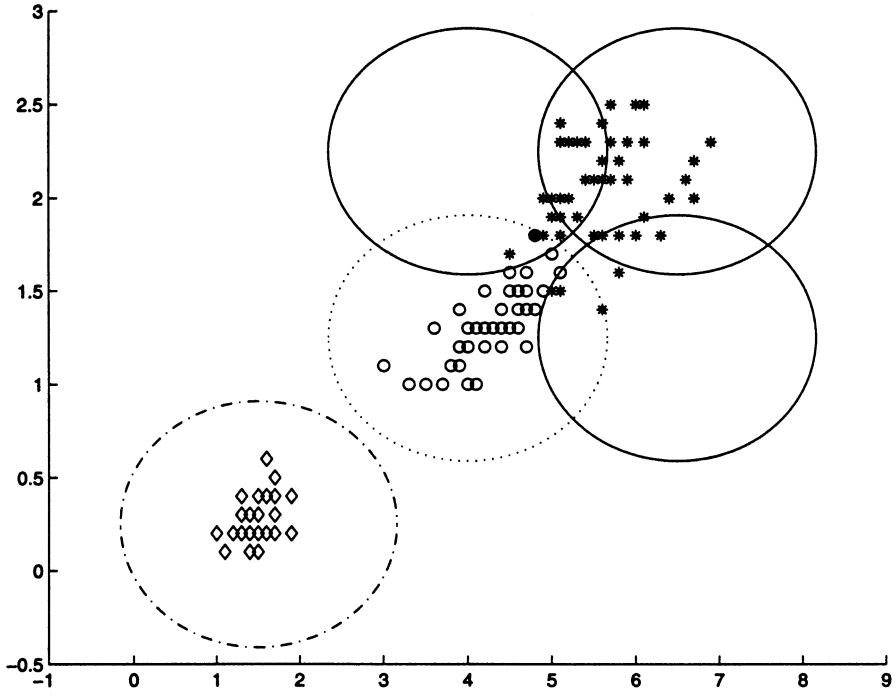

Fig. 6. The rules for classifying IRIS.

TABLE VIII
INITIAL ARCHITECTURE OF THE NETWORK USED FOR IRIS

Layer 1	3
Layer 2	12
Layer 3	81
Layer 4	3

TABLE IX
BEST RESUBSTITUTION ACCURACY FOR IRIS DATA FOR DIFFERENT RULE BASED CLASSIFIERS

Method	Rules	Resubstitution accuracy (%)
FuGeNeSys[46]	5	100
NEFCLASS[33]	7	96.7
ReFuNN[16]	9	95.3
EFuNN[18]	17	95.3
FeNe-I [9]	7	96.0
SuPFuNIS [43]	5	100

TABLE X
VALUE OF $1 - e^{-\beta_p{}^2}$ FOR DIFFERENT INPUT FEATURES FOR IRIS

Features	1	2	3	4
$1 - e^{-\beta_p^2}$	0.00	0.00	0.52	0.51

3) Results on PHONEME: Like IRIS, here also we used three fuzzy sets for each feature. Fig. 7 shows the membership functions used. In this case for each feature we used the same membership functions. The initial architecture of the network is presented in Table XII.

On termination of phase I, the number of misclassifications on X_{tr} was reduced to 85. The values of $1 - e^{-\beta_p{}^2}$ after phase I training (Table XIII) reveal that the network rejects only the fifth feature. The network is accordingly pruned for redundant nodes. In this case too, the network can retain its performance after pruning, i.e., produces the same misclassification of 85 on X_{tr}.

Pruning of redundant nodes reduces the number of antecedent clauses to 81. The removal of *incompatible rules* consequently yields 81 rules. Among these 81 rules 32 rules were *zero rules* and there were no *less used rules*. Therefore, the final network represented 49 rules. After phase III training the number of misclassifications on X_{tr} was 75(15%). This shows that tuning of membership function parameters enhances the performance of the classifier in this case. The misclassifications produced on X_{te} by the final network was 898(18.3%).

This data set has been extensively studied in [24] and the average misclassifications reported there on this data set using MLP are 15.40% on the training data and 19.63% on the test data. Using Radial Basis Function networks, the average training and test error were 19.9% and 22.48%, respectively [24]. Thus our results are quite comparable (in fact a little better) than the previously reported results.

4) Results on RS-Data: In all previous results reported, we selected the initial architecture of the network arbitrarily. Such initial networks may not yield good results in case of complex class structures. Also blind selection of initial networks may make the network too large and learning on such networks may become computationally very expensive. This problem becomes more severe when the number of features and number of classes are large. We demonstrate this problem in this example. We shall

TABLE XI
LINGUISTIC RULES FOR IRIS DATA

Rule no.	Rule
1	if pl is CLOSE TO 1.5 and pw is CLOSE TO 0.25 then class1
2	if pl is CLOSE TO 4.5 and pw is CLOSE TO 1.25 then class2
3	if pl is CLOSE TO 6.5 and pw is CLOSE TO 1.25 then class3
4	if pl is CLOSE TO 4.5 and pw is CLOSE TO 2.25 then class3
5	if pl is CLOSE TO 6.5 and pw is CLOSE TO 2.25 then class3

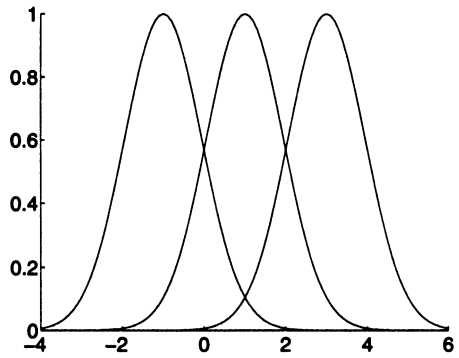

Fig. 7. The fuzzy sets used for all the five features of PHONEME.

TABLE XII
INITIAL ARCHITECTURE OF THE NETWORK USED TO CLASSIFY PHONEME

Layer 1	5
Layer 2	15
Layer 3	243
Layer 4	2

TABLE XIII
VALUE OF $1 - e^{-\beta_p{}^2}$ FOR DIFFERENT INPUT FEATURES FOR PHONEME

Features	1	2	3	4	5
$1 - e^{-\beta_p^2}$	0.66	0.68	0.98	0.93	0.00

TABLE XIV
VALUE OF $1 - e^{-\beta_p{}^2}$ FOR DIFFERENT INPUT FEATURES FOR RS-DATA

Features	1	2	3	4	5	6	7
$1 - e^{-\beta_p^2}$	0.99	0.00	0.99	0.41	0.28	0.00	0.98

TABLE XV
NUMBER OF FUZZY SETS FOR REDUCED RS-DATA

Feature 1	9
Feature 2	8
Feature 3	11
Feature 4	16
Feature 5	11

also discuss a methodology for deciding on the initial network. Most neuro-fuzzy techniques reported in literature do not pay adequate attention on the setting up of the initial network. The methodology that we discuss here is not a very general one, but it worked well with the present data set. More investigations are required in this area to evolve a general guideline for choosing the initial network.

The data set in question has seven input features and eight classes. With three fuzzy sets for each input feature, the number of antecedent nodes becomes 2187. After 100 iterations the values of the feature modulators almost stabilized and the values (as shown in Table XIV) suggest that features 2 and 6 are not important.

So, we discard these two unimportant features. Note that, the network is still quite big and also the use of only three fuzzy sets for each feature may not be adequate. So we now try to exploit some tools of exploratory data analysis to get a better initial network. The rest of the analysis is done on the remaining five features, we call this data set as REDUCED RS-DATA. For each class of the data we run the fuzzy c means (FCM) algorithm [3] with fuzzifier 2 and number of clusters 3. The number of clusters were determined in an *ad hoc* manner, but one may use some cluster validity index [3] to determine the "optimal" number of clusters. Thus, we obtained 3 prototypes for each class (in total 24 prototypes) in R^5. Let $\mathbf{p}_i = (p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5})^T$, where $i = 1$ to 24, be the prototypes. Let $X_i \subset X_{tr} \subset R^5$ be the set of data points represented by the prototype $\mathbf{p}_i$. Here X_i is obtained from the final partition matrix of the FCM algorithm. Let $\mathbf{x}_j^i = (x_{j1}^i, \ldots, x_{j5}^i)^T$; $j = 1$ to $|X_i|$ be the points in X_i. For each X_i and $k = 1$ to 5, we calculate

$$\sigma_{ik} = \frac{1}{|X_i|}\left[\sum_{j=1}^{|x_i|}\left(x_{jk}^i - p_{ik}\right)\right]^{\frac{1}{2}}. \tag{22}$$

For each feature k ($k = 1$ to 5), we take p_{ik} ($i = 1$ to 24), as the center and σ_{ik} as the spread of a bell shaped fuzzy membership function. In this way, for each feature we obtain 24 fuzzy sets. For each feature k, if $|p_{ik} - p_{jk}| \leq 2.0$, $i \neq j$ and if $\sigma_{ik} \geq \sigma_{jk}$, we discard the fuzzy set with center p_{jk}, otherwise we discard the fuzzy set with center p_{ik}. This is quite a natural choice as the peaks of two adjacent fuzzy sets should be at least two gray levels apart. The final number of fuzzy sets for each feature that we arrived at after this process is shown in Table XV.

Of all the rules represented by this network, we find only 132 antecedents producing a firing strength of more than 0.1 for more than three points. We retain only these 132 antecedents

TABLE XVI
PERFORMANCE AND RULE REDUCTION OF THE PROPOSED SYSTEM

Data set	Initial no. of rules	Final no. of rules	Misclassification on X_{tr}	Misclassification on X_{te}
ELONGATED	72	11	0%	0%
IRIS	243	5	3%	2%
PHONEME	486	49	15%	18%
RS-DATA	1056	91	18%	17%

which are supported by more than three data points. This drastically reduces the size of the network, yet there may be many redundant antecedents as we still allowed some weakly supported rules. We took this conservative approach with the hope that some of the weakly supported rules may become useful after tuning. So, we train a network with 132 antecedents (i.e., $132 \times 8 = 1056$ rules) discarding all other antecedents at the onset of training. After pruning of the incompatible rules we arrive at a network representing 132 rules. Out of these 132 rules we found 41 as less used. Thus, the final network represents 91 rules. This network gives a misclassification of 18.31% on the training data and 17.39% on the test data. RS-DATA was used by Kumar *et al.* [23] in a comparative study of different classification methods. The best results obtained by them using a fuzzy integral based scheme showed a misclassification of 21.85% on the test set. Our results with a reduced set of features outperform their results.

Table XVI summarizes the results on the various data sets used. It clearly shows a good performance of the proposed system.

VII. CONCLUSION

We proposed a new scheme for designing fuzzy rule based classifier in a neuro-fuzzy framework. The novelty of the system lies in in the fact that the network can select good features along with the relevant rules in an integrated manner. The network described here is also completely readable, i.e., one can easily read the rules required for the classification task from the network. The network starts with all possible rules and the training process retains only the rules required for classification, thus resulting in a smaller architecture of the final network. The final network has a lower running time than the initial network. The proposed method is tried on four data sets and in all four cases the network could select the good features and extract a small but adequate set of rules for the classification task. For one data set (Elongated) we obtained zero misclassification on both training and test sets and for all other data sets the results obtained are comparable to the results reported in the literature.

For further development of the methodology, some specialized tools of exploratory data analysis may be used to decide upon the number and definition of the input fuzzy sets. To some extent we did it in case of RS-DATA. But the method suggested to analyze RS-DATA needs further refinement and modifications to make it general in nature.

Another important problem of interest related to this may be to find the sensitivity of the output of a neuro-fuzzy classifier with respect to its internal parameters. The classifier described here is sensitive to the changes in the parameters of the membership functions and the certainty factors of the rules. We are currently working on this and hope to communicate our findings soon.

ACKNOWLEDGMENT

We are thankful to the referees and the Associate Editor for their valuable suggestions that made the manuscript more interesting.

REFERENCES

[1] E. Anderson, "The Irises of the Gaspe peninsula," *Bull. Amer. IRIS Soc.*, vol. 59, pp. 2–5, 1935.
[2] H. R. Barenji and P. Khedkar, "Learning and tuning fuzzy controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, 1992.
[3] J. C. Bezdek, J. Keller, R. Krishnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, MA: Kluwer, 1999.
[4] D. Chakraborty and N. R. Pal, "Integrated feature analysis and fuzzy rule-based system identification in a neuro-fuzzy paradigm," *IEEE Trans. Syst. Man Cybern. B*, vol. 31, pp. 391–400, 2001.
[5] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Int. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.
[6] ——, "Extracting fuzzy rules for pattern classification by cluster estimation," in *Proc. Sixth Int. Fuzzy Syst. Assoc. World Congress (IFSA '95)*, Sao Paulo, Brazil, July 1995, pp. 1–4.
[7] ——, "Extracting fuzzy rules from data for function approximation and pattern classification," in *Fuzzy Information Engineering*, D. Dubois, H. Prade, and R. R. Yagar, Eds. New York: Wiley, pp. 149–162.
[8] R. De, N. R. Pal, and S. K. Pal, "Feature analysis: neural network and fuzzy set theoretic approaches," *Pattern Recognition*, vol. 30, no. 10, pp. 1579–1590, 1997.
[9] S. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world applications," *Fuzzy Sets Syst.*, vol. 65, pp. 1–12, 1994.
[10] S. Haykin, *Neural Networks—A Comprehensive Foundation*. New York: Proc. Conc., 1994.
[11] H. Ishibuchi, T. Nakashima, and T. Morisawa, "Voting in fuzzy rule-based systems for pattern classification problem," *Fuzzy Sets Syst.*, vol. 103, pp. 223–238, 1999.
[12] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, 1995.
[13] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 297–307, 1988.
[14] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man. Cybern.*, vol. 23, pp. 665–685, 1993.
[15] J.-S. R. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proc. IEEE*, vol. 83, pp. 378–406, 1995.
[16] N. Kasabov, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems," *Fuzzy Sets Syst.*, vol. 82, pp. 135–149, 1996.
[17] N. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, 2002.

[18] N. Kasabov and B. Woodford, "Rule insertion and rule extraction from evolving fuzzy neural networks: algorithms and applications for building adaptive, intelligent expert systems," in *Proc. IEEE Int. Conf. Fuzzy Syst. FUZIEEE 99*, vol. 3, Seoul, Korea, Aug. 1999, pp. 1406–1411.
[19] J. Keller and Z. Chen, "Learning in fuzzy neural networks utilizing additive hybrid operators," in *Proc. Int. Conf. Fuzzy Logic and Neural Networks*, Iizuka, Japan, 1992, pp. 85–87.
[20] J. Keller and H. Tahani, "Implementation of conjunctive and disjunctive fuzzy logic rules in neural networks," *Int. J. Approx. Reasoning*, vol. 6, pp. 221–240, 1992.
[21] J. Keller, R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic," *Fuzzy Sets Syst.*, vol. 45, pp. 1–12, 1992.
[22] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic—Theory and Applications*. Englewood Cliffs, NJ: Prentice-Hall PTR, 1995.
[23] A. S. Kumar, S. Chowdhury, and K. L. Mazumder, "Combination of neural and statistical approaches for classifying space-borne multispectral data," in *Proc. ICAPRDT99*, Calcutta, India, 1999, pp. 87–91.
[24] L. Kuncheva, *Fuzzy Classifiers*: Physica-Verlag, 2000.
[25] R. Krishnapuram and J. Lee, "Propagation of uncertainty in neural networks," in *Proc. SPIE Conf. Robot. Computer Vision*, Bellingham, WA, 1988, pp. 377–383.
[26] ——, "Determining the structure of uncertainty management networks," in *Proc. SPIE Conf. Robot. Computer Vision*, Bellingham, WA, 1989, pp. 592–597.
[27] ——, "Fuzzy connective based hierarchical aggregation networks for decision making," *Fuzzy Sets Syst.*, vol. 46, no. 1, pp. 11–27, 1992.
[28] ——, "Fuzzy-sets-based hierarchical aggregation networks for information fusion in computer vision," *Neural Networks*, vol. 5, pp. 335–350, 1992.
[29] C. T. Lin and C. S. G. Lee, "Neural network based fuzzy logic control and decision system," *IEEE Trans. Computers*, vol. 40, pp. 1320–1335, 1993.
[30] ——, *Neural Fuzzy Systems*. Upper Saddle River, NJ: Prentice-Hall P T R, 1996.
[31] J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Trans. Neural Networks*, vol. 6, pp. 296–317, 1995.
[32] M. Mizumoto, "Pictorial representation of fuzzy connectives, part I: cases of t-norms, t-conorms and averaging operators," *Fuzzy Sets Syst.*, vol. 31, pp. 217–242, 1989.
[33] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets Syst.*, vol. 89, pp. 277–288, 1997.
[34] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification systems," *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 238–250, 1996.
[35] K. Pal, R. Mudi, and N. R. Pal, "A new scheme for fuzzy rule based system identification and its application to self-tuning fuzzy controllers," *IEEE Trans. Syst. Man Cybern. B*, vol. 32, pp. 470–482, 2002.
[36] K. Pal, N. R. Pal, and J. M. Keller, "Some neural net realizations of fuzzy reasoning," *Int. J. Intell. Syst.*, vol. 13, pp. 859–886, 1998.
[37] N. R. Pal, "Soft computing for feature analysis," *Fuzzy Sets Syst.*, vol. 103, pp. 201–221, 1999.
[38] N. R. Pal and C. Bose, "Context sensitive inferencing and "reinforcement-type" tuning algorithms for fuzzy logic systems," *Int. J. Knowledge-Based Intelligent Eng. Syst.*, vol. 3, no. 4, 1999.
[39] N. R. Pal and K. K. Chintalapudi, "A connectionist system for feature selection," *Neural, Parallel Scientif. Computat.*, vol. 5, no. 3, pp. 359–381, 1997.
[40] N. R. Pal, V. K. Eluri, and G. K. Mandal, "Fuzzy logic approaches to structure preserving dimensionality reduction," *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 277–286, 2002.
[41] N. R. Pal, K. Pal, J. C. Bezdek, and T. A. Runkler, "Some issues in system identification using clustering," in *Int. Joint Conf. Neural Networks, ICNN 1997*. Piscataway, NJ: IEEE, 1997, pp. 2524–2529.
[42] S. K. Pal and N. R. Pal, "Soft computing: goals, tools and feasibility," *J. IETE*, vol. 42, no. 4–5, pp. 195–204, 1996.
[43] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system," *IEEE Trans. Neural Networks*, vol. 13, pp. 578–599, 2002.
[44] D. W. Ruck, S. K. Rogers, and M. Kabrisky, "Feature selection using a multilayered perceptron," *J. Neural Network Computing*, pp. 40–48, 1990.
[45] M. Russo, "Genetic fuzzy learning," *IEEE Trans. Evolution. Computat.*, vol. 4, pp. 259–273.
[46] ——, "FuGeNeSys—a fuzzy genetic neural system for fuzzy modeling," *IEEE Trans Fuzzy Syst.*, vol. 6, pp. 373–387.
[47] M. Sugeno and T. Yasukawa, "A fuzzy-logic based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, 1993.
[48] M. Setnes and U. Kaymak, "Fuzzy modeling of client preference from large data sets: an application to target selection in direct marketing," *IEEE Trans. Fuzzy Syst.*, vol. 9, pp. 153–163, 2001.
[49] G. Tesauro and B. Janssens, "Scaling relationships in backpropagation learning," *Complex Syst.*, vol. 2, pp. 39–44, 1988.
[50] T. P. Volg, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the backpropagation method," *Biolog. Cybern.*, vol. 59, pp. 257–263, 1988.
[51] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, 1992.
[52] . [Online]ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA

Debrup Chakraborty received the B.E. degree in mechanical engineering from Jadavpur University, Calcutta, India, in 1997 and the M. Tech. degree in computer science from Indian Statistical Institute, Calcutta, in 1999.

He is currently working toward the Ph.D. degree with the Electronics and Communication Sciences Unit of the Indian Statistical Institute. His primary research interests include pattern recognition, neural networks, neuro-fuzzy integration schemes, and medical image analysis.

Nikhil R. Pal (M'91–SM'01) received the B.Sc. degree with honors in physics and the Master of Business Management degree from the University of Calcutta, India, in 1978 and 1982, respectively. He received the M.Tech. and Ph.D. degrees, both in computer science, from the Indian Statistical Institute, Calcutta, in 1984 and 1991, respectively.

Currently, he is a Professor with the Electronics and Communication Sciences Unit of the Indian Statistical Institute. His research interests include image processing, pattern recognition, fuzzy sets theory, measures of uncertainty, neural networks, evolutionary computation, and fuzzy logic controllers. He has coauthored a book *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Boston, MA: Kluwer Academic, 1999); co-edited two volumes titled "Advances in Pattern Recognition and Digital Techniques," ICAPRDT99 (Narosa, 1999) and "Advances in Soft Computing," AFSS 2002 (Springer-Verlag, 2002); and edited a book *Pattern Recognition in Soft Computing Paradigm* (Singapore: World Scientific, 2001).

Dr. Pal is an Associate Editor of the *International Journal of Fuzzy Systems*, *International Journal of Approximate Reasoning*, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—B. He is an area editor of *Fuzzy Sets and Systems*. He is a member of the editorial advisory board of the *International Journal of Knowledge-Based Intelligent Engineering Systems*, and a Steering Committee member of the journal *Applied Soft Computing*, Elsevier Science. He is an independent theme chair of the World Federation of Soft Computing and a governing board member of the Asia Pacific Neural Net Assembly. He was the Program Chair of the 4th International Conference on Advances in Pattern recognition and Digital Techniques, Dec. 1999, Calcutta, India, and the General Chair of the 2002 AFSS International Conference on Fuzzy Systems, Calcutta, 2002.