# Integrated Feature Analysis and Fuzzy Rule-Based System Identification in a Neuro-Fuzzy Paradigm

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

***Abstract*—Most methods of fuzzy rule-based system identification (SI) either ignore feature analysis or do it in a separate phase. This paper proposes a novel neuro-fuzzy system that can simultaneously do feature analysis and SI in an integrated manner. It is a five-layered feed-forward network for realizing a fuzzy rule-based system. The second layer of the net is the most important one, which along with fuzzification of the input also learns a modulator function for each input feature. This enables online selection of important features by the network. The system is so designed that learning maintains the nonnegative characteristic of certainty factors of rules. The proposed network is tested on both synthetic and real data sets and the performance is found to be quite satisfactory. To get an "optimal" network architecture and to eliminate conflicting rules, nodes and links are pruned and then the structure is retrained. The pruned network retains almost the same level of performance as that of the original one.**

***Index Terms*—Feature analysis, fuzzy systems, rule extraction, system identification.**

## I. Introduction

LET $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset R^s$ and $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \subset R^t$ and let there be an unknown function $\mathbf{S} : R^s \Rightarrow R^t$ such that $\mathbf{y}_k = \mathbf{S}(\mathbf{x}_k)\ \forall k = 1, \ldots, N$. In other words, there is an unknown function $\mathbf{S}$ which transforms $\mathbf{x}$ to $\mathbf{y}$. Given $X$ and $Y$, the problem of system identification (SI) is to find $S$ explicitly or implicitly. SI appears in various forms in science and engineering. There are many approaches to SI. Some models, like regression, are explicit in nature while others such as neural networks and fuzzy systems are computational transforms that do SI implicitly.

It is known that neural networks can act as universal approximators for a large class of nonlinear functions, hence the choice of neural networks for SI is quite justified and has been proved to be successful [5]. Neural networks are usually robust, possesses parallelism and good generalizing capabilities but they usually do not have readability and work as a black box. Hence, the underlying relation in a system, which has been approximated by a neural network, cannot be easily understood from the trained network by any easy means. On the other hand, fuzzy rule-based systems which have also been used for SI are highly interpretable in terms of linguistic rules. As fuzzy if-then rules can be easily understood by human beings and often an initial rule-base can be provided by an expert, there is no problem of readability. However, fuzzy rule-based systems, as such, are not capable of learning. Therfore, to extract the rules from a given data one has to depend on techniques like clustering or other tools of exploratory data analysis [30] or an initial rule base is supplied by an expert, which is then tuned using data. Thus, judicious integrations of neural networks and fuzzy logic are expected to result in systems with merits of both paradigms. Several attempts have been made to integrate fuzzy systems and neural networks with a view to achieving systems which are interpretable, robust, and have learning abilities [7], [21], [24]–[26], [33].

The various neuro-fuzzy unification schemes developed to date can be classified into three major groups:

1) neural fuzzy systems;
2) fuzzy neural systems;
3) cooperative systems.

Neural fuzzy systems are fuzzy systems implemented by neural networks [10], [11], [24], [25], [31], [33]. Fuzzy neural systems are neural networks capable of handling fuzzy information [4], [6], [33]. The inputs, outputs, and weights of fuzzy neural networks could be fuzzy sets, often fuzzy numbers or membership values. The cooperative systems are those which use different paradigms (neuro or fuzzy) to solve various facets of the same problem [33]. All three of these paradigms taken together is known as *neuro-fuzzy computing*. The scheme that we are going to present here is a neural fuzzy system. Hence to begin with we discuss some previous attempts in this direction.

Lee *et al.* [20] proposed a neural network model for fuzzy inferencing. They developed an algorithm for adjusting (tuning) the membership functions of antecedent linguistic values of the rule set by error back-propagation (EBP), where the consequent parts were considered fixed. Li and Wu [22] proposed a neuro-fuzzy hierarchical system with if-then rules for pattern classification problem. A five-layer network is also presented in [39]. The parameters of the net are identified using evolutionary programming and the tuned network is then pruned to extract a small set of rules. Lin and Lee [24] presented a multilayered feedforward connectionist model designed for fuzzy logic control and decision making. A hybrid two-step learning scheme that combined self-organized (unsupervised) and supervised learning algorithms for selection of fuzzy rules and tuning of membership functions were developed. Lin and Lee used Kohonen's self-organizing feature map [15] for finding the centers of the membership functions. After selection of the rule set, i.e., when the network architecture is established, the second step of supervised learning begins. Some heuristic guidelines for rule reduction and combination

Manuscript received August 4, 2000; revised January 22, 2001. This work was supported in part by the Department of Science and Technology, Government of India under Grant III-5(21) 96-97. This paper was recommended by Associate Editor L. O. Hall.

The authors are with Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700035, India (e-mail: debrup_r@isical.ac.in; nikhil@isical.ac.in).

Publisher Item Identifier S 1083-4419(01)05663-1.

were also provided. Shann and Fu [35] presented a layered network for selection of rules. Initially, the network was constructed to contain all possible fuzzy rules. After EBP training, the redundant rules were deleted by a rule pruning process for obtaining a concise rule base. The architecture of Shann and Fu is similar to that of Lin and Lee in several respects. Pal and Pal [32] discussed some limitations of the scheme by Shann and Fu and provided a better rule tuning and pruning strategy. Lin and Cunningham [27] also developed a layered network for SI. They used fuzzy curves for feature selection, but this phase was a part of preprocessing on the data before the data got into the network. Wu and Er [38] proposed a dynamic fuzzy neural network implementing Takagi-Sugeno-Kang fuzzy system based on extended radial basis function network. Lin and Chung [23] developed a neuro-fuzzy combiner based on reinforcement learning for multiobjective control. Figureiredo and Gomide [3] proposed a neural fuzzy system which encodes the knowledge learned in the form of fuzzy if-then rules and processes data using fuzzy reasoning principles. After learning linguistic rules can be easily extracted from the network. Kim and Kasabov [12] developed a neuro-fuzzy inference system which consists of two phases, one of rule generation from data and a rule tuning phase by EBP. Krishnapuram and Lee developed a neural network for classification which uses fuzzy aggregation functions as activation functions [18], [19]. On completion of training, the redundant links can be identified and removed from the net. If all links emanating from an input node are removed, then the corresponding feature is redundant and hence eliminated. Similar types of networks are also discussed in [8], [9], [16], and [17].

Most of the methods discussed here do not *explicitly* do feature analysis. However, it is well known that feature analysis plays an important role in SI [28], [36]. For example, consider a system with input $\mathbf{x} \in R^s$ and output $\mathbf{y} \in R^t$. It may be possible that not all the $s$ input features are required to understand the relation between the input and output or, may be some of the features are redundant or indifferent to the output of the system. Moreover, more features are not necessarily good, some features may even have derrogatory effect on the output. Thus, selection of an appropriate subset of features, for the given task at hand, not only can reduce the cost of the system but also may improve the performance of the system.

There are many methods of feature analysis or feature ranking. Details of some of the feature analysis methods using soft computing tools like fuzzy logic, neural networks, and genetic algorithms can be found in [2] and [28]. Several authors have also designed neural networks that simultaneously learn feature extraction and classification. For example, Won *et al.* proposed a shared weight morphological network and applied it to target detection in [37]. Similar kinds of work can be found in [1] and [13].

Following the concept of Pal and Chintalapudi [29], the feature selection scheme proposed here uses a modulator function. Pal and Chintalapudi used a multilayered feed-forward architecture. Every input feature was multiplied by an attenuation function prior to its entry in the network. The attenuation functions were so designed that they took values between 0 and 1. The parameters of the attenuation functions were learned by the EBP learning scheme. After training, for a bad or indifferent feature, the attenuation function acquires a value close to 0 and for a good feature a value close to 1. The present work is inspired by the feature selection scheme of Pal and Chintalapudi but the present philosophy and formulation used here are quite different.

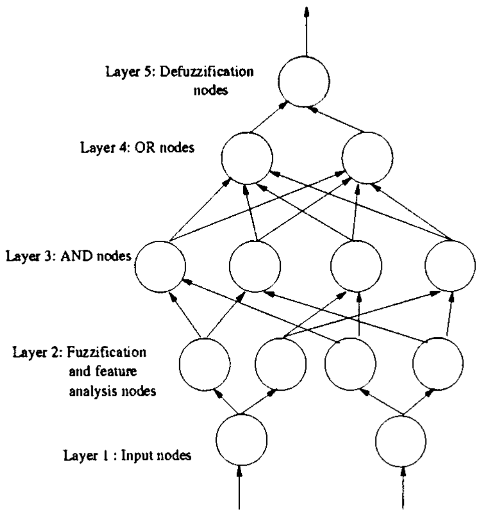
Layer 5: Defuzzification nodes
Layer 4: OR nodes
Layer 3: AND nodes
Layer 2: Fuzzification and feature analysis nodes
Layer 1 : Input nodes

Fig. 1. Network structure.

Here, we present a neural fuzzy system for simultaneous feature selection and SI. In subsequent sections, we discuss the network structure of the proposed system followed by the learning rules, optimization of the network, and some simulation results. Finally, the paper is concluded in Section VII, which also gives some directions of future works on the proposed system.

## II. The Network Structure

Let there be $s$ input features $(x_1, x_2, \ldots, x_s)$ and $t$ output features $(y_1, y_2, \ldots, y_t)$. The proposed neural fuzzy system will deal with fuzzy rules of the form $R_i$: If $x_1$ is $A_{1i}$ and $x_2$ is $A_{2i}$ … and $x_s$ is $A_{si}$ then $u_{ji}$ is $B_{ji}$. Here, $A_{ji}$ is the $i$th fuzzy set defined on the domain of $x_j$ and $B_{ji}$ is the $i$th fuzzy set defined on the domain of $y_j$.

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 $A_{ij} = A_{ik}$ for some $j$ and $k$. Similar is the case for the linguistic values defined on the output variables.

The neural fuzzy system is realized using a five-layered network, as shown in Fig. 1. The node functions with its inputs and outputs are discussed layer by layer. We use suffixes $p$, $n$, $m$, $l$, and $k$ to denote, respectively, the suffixes of the nodes in layers 1 through 5 in order. 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 representing its linguistic values. Thus, the number of nodes in this layer is equal to the number of input features in the data. If

$x_p$ denotes the input to any node in layer 1 then the output of the node will be

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

*Layer 2:* Each node in layer 2 represents the membership functions of a linguistic value associated with an input linguistic variable. Moreover, this layer also does the feature analysis. The output of these nodes lies in the interval [0,1] and represents the membership grades of the input with respect to different linguistic values. Therefore, the nodes in this layer acts as fuzzifiers. The most commonly used membership functions are triangular, trapezoidal and bell shaped. Although any one of these choices may be used, we consider bell shaped membership functions. All connection weights between the nodes in layer 1 and layer 2 are unity. If there are $N_i$ fuzzy sets associated with the $i$th feature and if there are $s$ input features then the number of nodes in this layer would be $N^2 = \sum_{i=1}^{s} N_i$. The output of a node in layer 2 is denoted by

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

In (2), the subscript $n$ denotes the $n$th term (fuzzy set) of the linguistic variable $x_p$. $\mu_n$ and $\sigma_n$ represent the mean and spread respectively of the bell shaped function representing a term of the linguistic variable $x_p$ associated to node $n$ .

For the purpose of feature selection, the output of this layer needs to be modified so that every indifferent/bad feature $x_p$ gets eliminated. If a linguistic variable $x_p$ is not important (or is indifferent) for describing the system behavior, i.e., for defining the input-output relation, then the values of $x_p$ should not have any effect on the firing strength of the rules involving that input variable. This is our main guiding principle for feature analysis and it makes our approach completely different from the work of Pal and Chintalapudi [29]. Since for any $T$-norm, $T(1, \alpha) = \alpha$, $0 \le \alpha \le 1$, this can be realized if an indifferent feature always generates a membership of unity. This may appear impossible at the first sight. Note that for an indifferent feature, all of its terms (i.e., all of its linguistic values) should have no effect on the firing strength. Next we explain how this can be realized.

Let us associate a function $f_n$ with each node $n$ in layer 2. We call $f_n$ a modulator function. For an indifferent (or bad) feature we want all linguistic values defined on that feature to result in a membership value of 1. To achieve this, we model $f_n$ as

$$f_n = \exp\left[\lambda_p \ln\left(\frac{1}{\bar{z}_n}\right)\right]. \tag{3}$$

Here, $\lambda_p \in [0, 1]$ is a parameter associated with a particular *linguistic variable* $x_p$ of which node $n$ is a term. From (3) we see that when $\lambda_p$ is nearly 1 then $f_n$ is nearly $1/\bar{z}_n$, and when $\lambda_p$ is nearly 0 then $f_n$ is nearly 1. Therefore, for bad features $\lambda_p$ should get large values (close to 1) and small values (close to 0) for good features. Thus, for a bad feature, the modulated membership value would be $f_n \cdot \bar{z}_n \approx \bar{z}_n \cdot (1/\bar{z}_n) \approx 1$ irrespective of the value of $x_p$. Similarly, for a good feature, the modulated membership value would be $f_n.\bar{z}_n \approx 1.\bar{z}_n \approx \bar{z}_n \approx$ the actual membership value. Since $\lambda_p$ must take values between 0 and 1, we model $\lambda_p$ by $e^{-\beta_p^2}$. Thus, the activation function of any node $n$ in layer 2 would be as

$$z_n = \bar{z}_n \exp\left[e^{-\beta_p^2} \ln\left(\frac{1}{\bar{z}_n}\right)\right] \tag{4}$$

which can be simplified to

$$z_n = \bar{z}_n^{(1-e^{-\beta_p^2})} \tag{5}$$

where $\bar{z}_n$ is computed using (2). The parameter $\beta_p$ can be learned by back-propagation or by some other technique. We see that when $\beta_p^2$ takes a large value then $z_n$ tends to $\bar{z}_n$ and for small values of $\beta_p^2$, $z_n$ tends to 1, thereby making the feature indifferent. Therefore, our objective would be to make $\beta_p^2$ take large values for good features and small values for bad ones through the process of learning. Layer 2 can be better realized using two layers of neurons, the first one for computation of the membership value, $\bar{z}_n$ and second layer for the modulated output using (5).

*Layer 3:* This layer is called the AND layer. Each node in this layer represents an IF part of a fuzzy rule. There are many operators ($T$-norms) for fuzzy intersection [14]. Here, we choose product as the operator for intersection. The number of nodes in this layer is $N^3 = \prod_{i=1}^{s} N_i$. The output of the $m$th node in the layer is

$$z_m = \prod_{n \in P_m} z_n \tag{6}$$

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

*Layer 4:* This is the OR layer and it represents the THEN part (i.e., the consequent) of the fuzzy rules. The operation performed by the nodes in this layer is to combine the fuzzy rules with the same consequent. 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 AND node $m$ in layer 3 as the IF part and the OR node $l$ in layer 4 representing the THEN part. These weights are adjustable while learning the fuzzy rules. If there are $M_i$ fuzzy sets associated with the $i$th output variable and there are $t$ output features then the number of nodes in this layer is $N^4 = \sum_{i=1}^{t} M_i$. For simplicity let us assume that there is only one output variable and $M$ linguistic values are defined on it. Therefore, the fourth layer has $N^4 = M$ nodes. For each output linguistic value there are exactly $N^3$ rules having that value as the consequent. Every node of this layer picks up only one rule from among the associated $N^3$ rules based on the maximum agreement with facts (in terms of the product of firing strength and certainty factor) for computation of the defuzzified output. When all certainty factors are equal, the rules are selected based on the maximum firing strength. This rule selection is viewed as an OR operation and realized by the $\max$ operator. Thus, like Shann and Fu [35] and Pal and Pal [32], the output of the node $l$ in layer 4 is computed by

$$z_l = \max_{m \in P_l} (z_m w_{lm}) \tag{7}$$

where $P_l$ represents the set of indexes of the nodes in layer 3 connected to the node $l$ of layer 4. Since the learnable weights $w_{lm}$'s are interpreted as certainty factors, each $w_{lm}$ should be nonnegative. The EBP 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 $g_{lm}^2$. The $g_{lm}$ is unrestricted in sign but the effective weight $w_{lm} = g_{lm}^2$ will always be nonnegative. Therefore, the output (activation function) of the $l$th node in layer 4 will be

$$z_l = \max_{m \in P_l}(z_m g_{lm}^2). \tag{8}$$

*Layer 5:* This layer is the defuzzification layer. Each node of layer 5 represents an output linguistic variable and performs defuzzification, taking into consideration the effects of all membership functions of the associated output linguistic variable. The number of nodes in this layer is equal to the number of output features. Here, we use the centroid defuzzification scheme, and a node in this layer computes the output as

$$z_k = \frac{\sum_{l \in P_k} z_l a_l c_l}{\sum_{l \in P_k} z_l a_l}. \tag{9}$$

In (9), $P_k$ is the set of indexes of the nodes in layer 4 connected to node $k$ in layer 5 and $a_l, c_l$ are the spread and mean of the membership function representing node $l$ in layer 4. The weights of the links connecting nodes in layer 4 and layer 5 are unity.

## III. Learning of Feature Modulators and Rules

We now derive the learning rules for the neural fuzzy system with the activation or node functions described in the previous section. In the training phase, the concept of back-propagation is used to minimize the error function

$$e = \frac{1}{2}\sum_{i=1}^{N} E_i = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{t}(y_{ik} - z_{ik})^2 \tag{10}$$

where $t$ is the number of nodes in layer 5 and $y_{ik}$ and $z_{ik}$ are the target and actual outputs of node $k$ in layer 5 for input data $\mathbf{x}_i$; $i = 1, 2, \ldots, N$. The method for adjusting the learnable weights in layer 4 and the parameters $\beta_p$ in layer 2 are 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, we drop the subscript $i$ in our subsequent discussions.

The delta value $\delta$ of a node in the network is defined as the influence of the node output with respect to $E$. The derivation of the delta values and the adjustment of the weights and the parameters $\beta_p$ are presented layer wise next.

*Layer 5:* The output of the nodes in this layer is given by (9) and $\delta$ values for this layer, $\delta_k$, will be

$$\delta_k = \frac{\partial E}{\partial z_k}.$$

Thus

$$\delta_k = -(y_k - z_k). \tag{11}$$

*Layer 4:* The delta for this layer would be

$$\delta_l = \frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial z_l}.$$

In other words,

$$\delta_l = \delta_k \frac{a_l(c_l - z_k)}{\sum_{l' \in P_k} z_{l'} a_{l'}} \tag{12}$$

where $k$ is a node in layer 5 with which node $l$ in layer 4 is connected.

*Layer 3:* The delta for this layer would be

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

Hence, the value of $\delta_m$ will be

$$\delta_m = \begin{cases} \sum_{l \in Q_m} \delta_l\, g_{lm}^2, & \text{if } z_m\, g_{lm}^2 = \max_{m'}\{z_{m'} g_{lm'}^2\} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

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 $\delta_n$ for layer 2 would be

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

Hence,

$$\delta_n = \sum_{m \in R_n} \delta_m \left(\frac{z_m}{z_n}\right). \tag{14}$$

In (14), $R_n$ is the set of indexes of nodes in layer 3 connected with node $n$ in layer 2.

With the $\delta$ calculated for each layer now we can derive the weight updating equation and the equation for updating $\beta_p$

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

or

$$\frac{\partial E}{\partial g_{lm}} = \begin{cases} \sum_{l \in Q_m} 2\delta_l\, z_m\, g_{lm}, & \text{if } z_m\, g_{lm}^2 = \max_{m'}\{z_{m'} g_{lm'}^2\} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Similarly, we calculate

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

or

$$\frac{\partial E}{\partial \beta_p} = -\sum_{n \in R_p} \delta_n \left(2\,\beta_p\, e^{-\beta_p^2}\, z_n\right)\left(\frac{z_p - \mu_n}{\sigma_n}\right)^2 \tag{16}$$

where $R_p$ is the set of indexes of nodes in layer 2 connected to node $p$ of layer 1. Hence, the update equations for weights and $\beta_p$ are

$$g_{lm}(t+1) = g_{lm}(t) + \eta\left(-\frac{\partial E}{\partial g_{lm}}\right) \tag{17}$$

and

$$\beta_p(t+1) = \beta_p(t) + \mu\left(-\frac{\partial E}{\partial \beta_p}\right). \tag{18}$$

In (17) and (18), $\eta$ and $\mu$ are learning coefficients.

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 $\beta$'s are so selected that no feature gets into the network in the beginning. This is realized by assigning very low positive values (say, 0.0001) to each $\beta_p$. Thus in the beginning of learning every node in layer 2 produces a value which is nearly equal to one, and consequently, all features are considered unimportant. As learning proceeds, the values of $\beta$'s gets updated in such a way that the important features, i.e., the features which can reduce the error rapidly, only pass through the network. Next, we discuss strategies to prune the network to get an "optimal" readable network.

## IV. Optimizing the Network

We started with a network which represented all possible rules given a set of input and output fuzzy sets. But all possible rules usually are never needed to represent a system. Moreover, the modulator functions associated with the second layer, may decide that all of the features are not important. Hence, some of the nodes present in the network may be redundant, and presence of these redundant nodes will decrease the readability/interpretability of the network. We know that a SI task can be easily handled by a conventional multilayered perceptron (MLP) network, but we have used a neural fuzzy system for the purpose of SI to increase the readability of the network, so that we can understand the relation between the inputs and outputs in terms of linguistic rules. Thus, to make the network optimal and more readable, we need to prune it removing redundant nodes and incompatible rules. We next discuss what we mean by redundant nodes and incompatible rules, and how to remove them.

### A. Pruning Redundant Nodes

Let us consider an SI problem with $s$ input features so that layer 1 of the network will have $s$ nodes. Let the indexes of these nodes be denoted by $p$ ($p = 1$ to $s$). Let $\mathcal{N}_p$ be the set of indexes of the nodes in layer 2 which represents the fuzzy sets on the feature represented by node $p$ of layer 1 and let $|\mathcal{N}_p| = N_p$. We also assume that $c\,(c < s)$ of the $s$ features are indifferent/bad as dictated by the training. Let $R$ be the set of indexes of the nodes which represents the $c$ indifferent/bad features. Hence, any node with index $p$ in layer 1 such that $p \in R$ is redundant. Also, any node $n$ in layer 2, where $n \in \mathcal{N}_p$ and $p \in R$, is also redundant. In our network construction, a node in layer 3, can be uniquely identified by its connections with the nodes in layer 2. We can indicate a node $m$ in layer 3 as $S_m = [x_{m1}, x_{m2}, \ldots, x_{ms}]$ where $x_{mp} \in \mathcal{N}_p$. Now for any $p \in R$ we can group the nodes in layer 3 into $N_p$ many groups, we call them $G_{pr}$, where $r = 1, 2, \ldots, N_p$. Every node in the $r$th group is connected to the $r$th fuzzy set on the $p$th feature. Let $S_m$ be a node in layer 3 which belongs to the $r$th group, i.e., $S_m = [x_{m1}, x_{m2}, \ldots, x_{ms}] \in G_{pr}$. Then, for every group $G_{p\kappa}$, $\kappa \neq r$, $s = 1, 2, \ldots, N_p$, there exists exactly one node $S_q = [x_{q1}, x_{q2}, \ldots, x_{qs}]$, such that $x_{qj} = x_{mj}, \forall j \neq p, j = 1, 2, \ldots, s$, where $p \in R$ is a bad feature.

Thus, every group of nodes has identical connection structure with the nodes of layer 2 except for its connection to a node corresponding to the redundant feature $p$, and as per our construction that particular node produces an output membership value of 1, for all feature values. Hence, in layer 3 it is enough to keep only one of the $N_p$ groups and the other $N_p - 1$ groups of nodes are redundant.

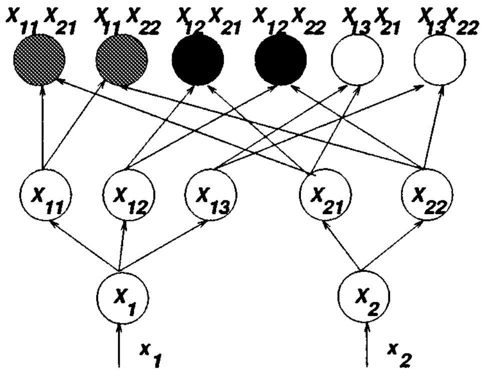

Fig. 2. Subnet to illustrate redundant nodes.

To elucidate the concept of redundant nodes, let us consider an SI task with two input features $x_1$ and $x_2$. Therefore, layer 1 of the network for this task will have two nodes, we name them as $X_1$ and $X_2$ Fig. 2. We also assume that input feature $x_1$ has three fuzzy sets associated with it and the feature $x_2$ has two fuzzy sets associated with it. Hence, layer 2 will have three nodes, $X_{11}$, $X_{12}$, and $X_{13}$, connected with $X_1$, and two nodes, $X_{21}$ and $X_{22}$, connected to $X_2$. The nodes in layer 3 are named using their connections to nodes in layer 2, e.g., a layer 3 node connected to $X_{11}$ and $X_{22}$ will be denoted by $X_{11}X_{22}$ (Fig. 2). Now, if, training dictates feature $x_1$ to be redundant then irrespective of the values of $x_1$, each of the nodes $X_{11}$, $X_{12}$, and $X_{13}$ will produce an output of unity. In this case we can group the nodes in layer 3 into three groups, which are shown by white nodes, gray nodes, and black nodes in Fig. 2. Since $X_{11}$ produces an output of unity, the gray group has two nodes representing two antecedent clauses "$x_2$ is $X_{21}$" and "$x_2$ is $X_{22}$." Similarly, each of the white and black groups also represents the same two antecedent clauses as the outputs of both $X_{12}$ and $X_{13}$ are 1. Hence, it is enough to retain any one of the three groups. Note that in this case, if two group of nodes are pruned, then the third layer looses its importance, as it really does not do any AND-ing operation. But such a situation, will rarely occur where out of only two input features one is redundant. If it happens, then the third layer simply transfers its input to the next layer.

The redundant nodes are not required for the SI task, but they add to the computational overhead of the network. Therefore, removal of these nodes is necessary to get an optimal network. The crucial part of this method is determination of the set of redundant nodes in layer 1. For this we use the value of $1 - e^{-\beta_p^2}$ (we call it as $\gamma_p$) as an indicator. We have seen earlier, that for good features $\gamma_p$ takes values close to 1 and for bad features it is close to 0. Therefore, we fix a small positive threshold $th$ such that $p \in R$ if $\gamma_p < th$. Next, we summarize the method of the removal of redundant nodes (here the removal of a node also means removal of its incoming and outgoing links) followed by a discussion on the choice of threshold $th$.

```
Algorithm: Pruning of redundant nodes
begin
  $R = \phi$
   for each $p$ in layer 1
     if ($\gamma_p < th$)
       $R = R \cup \{p\}$
       remove node $p$ in layer 1
       remove the nodes in layer 2 connected to $p$
     end if
   end for
  do while ($R \neq \phi$)
     let $i \in R$
     $R = R - \{i\}$
    find $G_{ij}$, $j = 1, 2, \ldots, N_i$
    remove nodes in $G_{i2}, G_{i3}, \ldots, G_{iN_i}$
   end do
end
```

1) *Selection of the Threshold $th$:* Here, we present a guideline for selecting the threshold $th$. We have used Gaussian membership functions for the input fuzzy sets (also for output fuzzy sets), hence as per our formulation the output of the nodes in layer 2 can be represented by

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

where

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

and $\gamma_p = 1 - e^{-\beta_p^2}$. If we consider $\sigma_n = \sqrt{2}\sigma_n{}'$, then we have

$$\bar{z}_n = \exp\left\{-\frac{(z_p - \mu_n)^2}{2\sigma'^2_n}\right\} \tag{19}$$

and

$$z_n = \left[\exp\left\{-\frac{(z_p - \mu_n)^2}{2\sigma'^2_n}\right\}\right]^{\gamma_p}. \tag{20}$$

We know that 99% of the area under the membership function in (19) lies over the interval $[\mu_n - 3\sigma'_n, \mu_n + 3\sigma'_n]$. Consequently, the value of $\bar{z}_n$, beyond this interval would be negligibly small. For a bad/indifferent feature we want the modulated membership value $z_n$ to be almost unity over the entire interval $[\mu_n - 3\sigma'_n, \mu_n + 3\sigma'_n]$. Therefore, we can safely choose that value of $\gamma_p$ as the threshold $th$, which makes $z_n = c$ ($c \approx 1$) at $z_p = \mu_n - 3\sigma'_n$ and at $\mu_n + 3\sigma'_n$. Thus, from (20), we obtain the threshold $th = -\ln(c)/4.5$. Note that for such a choice if $z_p \in (\mu_n - 3\sigma'_n, \mu_n + 3\sigma'_n)$, then $z_n \geq c$. If we consider $c = 0.8$, then we obtain $th = 0.05$, which we use in the simulations.

### B. Pruning of Incompatible Rules

According to our construction of the network, the links between layer 3 and layer 4 represent the rules, and the weights associated with the links can be interpreted as the certainty factor

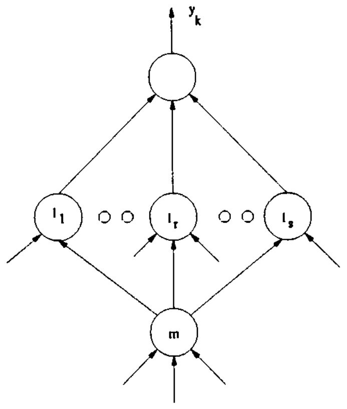

Fig. 3. Incompatible rules.

of the rules. But as the nodes in layer 3 and layer 4 are fully connected, initially, all fuzzy rules are considered. If there are $\tau$ linguistic values for an output linguistic variable then there are $\tau$ rules with the same antecedent but different consequents, which are inherently inconsistent. Let us consider the subnet in Fig. 3, which shows only the connections used for selecting the most relevant rule corresponding to the antecedent clause (IF part) represented by the node $m$ in layer 3. Fig. 3 corresponds to the following incompatible rules.

If $(antecedent)_m$ then $y_k$ is $T_{l,z_k}(w_{k_r m})$, $l = 1, 2, \ldots, \tau$.

Where, $(antecedent)_m$ is the antecedent clause represented by node $m$ of layer 3, $T_{l,z_k}$ is the $l$th fuzzy set on the $k$th output variable $y_k$. The certainty factors $w_{k_r m}$ of the rules are shown in parenthesis.

For rule pruning the centroid of the set of incompatible rules is calculated considering the connections in Fig. 3 as

$$c_{km} = \frac{\sum_{l \in P_k} z_l a_l c_l}{\sum_{l \in P_k} z_l a_l}. \tag{21}$$

Since $z_l = z_m g_{lm}^2$

$$c_{km} = \frac{\sum_{l \in P_k} z_m g_{lm}^2 a_l c_l}{\sum_{l \in P_k} z_m g_{lm}^2 a_l}. \tag{22}$$

Hence

$$c_{km} = \frac{\sum_{l \in P_k} g_{lm}^2 a_l c_l}{\sum_{l \in P_k} g_{lm}^2 a_l}. \tag{23}$$

$c_{km}$ in (23) can be viewed as a centroid of the set of incompatible fuzzy rules which corresponds to Fig. 3 with certainty factor $g_{lm}^2$ for the rule with antecedent node $m$ and consequent node $l$. We calculate the membership values of $c_{km}$ in all consequent fuzzy sets of the incompatible rules. Then the rule which have the highest membership value for $c_{km}$ is selected and the other rules are deleted.

## V. Training Phases

The training of the system takes place in three phases. Phase 1 is called the feature selection phase, where the training is done on the initial network with all the possible nodes and links. The

Phase 1 training is considered to be over once the modulator functions stabilize, i.e., when

$$\frac{||\mathbf{\Gamma}(t) - \mathbf{\Gamma}(t+1)||}{s} < \epsilon$$

where $\mathbf{\Gamma}(t) \in R^s$ is the vector of $\gamma_p$ values after the $t$th epoch and $\epsilon$ is a small positive constant.After Phase 1 training is over, based on the values of the parameter $\gamma$, the pruning of the redundant *nodes* is done. After pruning, the output of the second layer nodes would be as

$$z_n = \bar{z}_n$$

as the modification of the membership value for the purpose of feature selection will no longer be required. After pruning, the network is retrained for a few epochs to adapt its weights in its new reduced architecture, and this phase is called Phase 2 of training. Finally, the incompatible rules (*links*) are pruned and again the network is allowed to learn in its new architecture, which is termed as Phase 3 of training. Let $\mathbf{W}(t)$ denote the vector of the weights of all the links connecting layer 3 and layer 4 after the $t$th epoch in Phase 2. The Phase 2 training can now be stopped when

$$\frac{||\mathbf{W}(t) - \mathbf{W}(t+1)||}{|\mathbf{W}(t)|} < \epsilon$$

where $|\mathbf{W}(t)|$ gives the number of components in $\mathbf{W}(t)$. Phase 3 tuning can also be terminated based on the same criteria. Note that for Phase 3, the number of components of $\mathbf{W}(t)$ will be less than that in Phase 2. However in the present simulations we have arbitrarily chosen the number of epochs. After the Phase 3 training is over, we obtain a network which is readable, and the rules that describe the input-output relation can be easily retrieved from the final architecture of the network.

## VI. Results

The methodology developed is tested on two data sets taken from [36] and the performance is found to be quite satisfactory. We first describe the data sets and then in two separate subsections we present the results obtained on them.

Of the two data sets one is synthetically generated and the other is a real life one. The first one is named HANG which is generated by

$$y = (1 + x_1^{-2} + x_2^{-1.5})^2, \quad 0 \leq x_1, x_2 \leq 5. \tag{24}$$

The graph of (24) is shown in Fig. 4. Equation (24) represents a nonlinear system with two inputs $x_1$ and $x_2$ and a single output $y$. We randomly took 50 points from $0 \leq x_1, x_2 \leq 5$ and obtained 50 input-output data points according to (24). To illustrate the feature analysis capability of the proposed net, we added two random variables $x_3$ and $x_4$, in the range [0,5] as dummy inputs. It is expected that features $x_3$ and $x_4$ would be indifferent to the output of the system.

The second data set is called CHEM. This is the data for operator's control of a chemical plant for producing a polymer

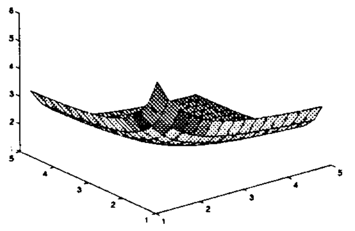

Fig. 4. Plot of HANG.

by polymerization of some monomers. There are five input features, which a human operator may refer to for control and one output, that is his/her control. The input variables are monomer concentration ($u_1$), change of monomer concentration ($u_2$), monomer flow rate ($u_3$), two local temperatures inside the plant ($u_4$, and $u_5$). The only output ($y$) is the set point for monomer flow rate. In [36], there is a set of 70 data points obtained from an actual plant operation. We name this data set as CHEM and use as our training data. In [36], it has been reported that the two local temperatures inside the plant, i.e., $u_4$ and $u_5$ *do not significantly contribute* to the output.

One of the most important issues for rule-based SI is to determine the input and output fuzzy sets. We do not use any sophisticated technique in this regard. We found out the domain of each input and output component and picked up a number of fuzzy sets to span the whole range with considerable overlap between adjacent fuzzy sets. As stated earlier we used fuzzy sets with Gaussian membership functions.

We measure the performance of our system by the sum of squared errors (SSE) and maximum deviation (MD) of the output from the target. Lin and Cunningham [27] defined a performance index (PI) as

$$PI = \frac{\sqrt{\sum_{k=1}^{N}(z_k - y_k)^2}}{\sum_{k=1}^{N}|y_k|} \tag{25}$$

where $z_k$ denotes the output at an output node $k$ and $y_k$ denotes the desired output at the same node. However, in [34], it was pointed out that this PI is monotonically decreasing with $N^{-1/2}$, i.e., it is possible to obtain a very small PI just by increasing $N$. Still, we evaluated the performance of our system based on PI in (25) for the sake of easy comparison.

### A. Results on HANG

Here, we used four input fuzzy sets for each input feature and five output fuzzy sets for the output linguistic variable. The input and output fuzzy sets are shown in Figs. 5 and 6, respectively. Hence, the initial architecture for this problem is as described in Table I.

The network was trained using the data set, HANG with learning parameters $\eta = 0.1$ and $\mu = 0.1$ for 1000 epochs in Phase 1, 500 epochs in Phase 2, and 3500 epochs in Phase 3. The SSE was reduced from 57.907 to 1.513. The PI was equal to 0.01, which is comparable to the result obtained by Sugeno and Yasukawa [36], who obtained a PI of 0.01. Using this data, Lin and Cunningham [27] obtained a PI of 0.003, but in their case they used only the good features, i.e., only features $x_1$ and $x_2$. Moreover, we did not tune the membership functions

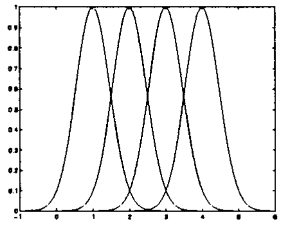

Fig. 5. Input membership functions used for HANG.

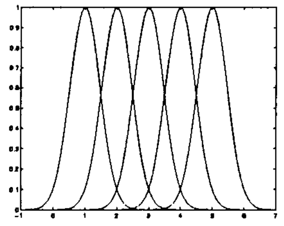

Fig. 6. Output membership functions used for HANG.

TABLE I
ARCHITECTURE OF THE NEURAL FUZZY SYSTEM USED FOR HANG

| layer no. | no. of nodes |
|---|---|
| 1 | 4 |
| 2 | 16 |
| 3 | 256 |
| 4 | 5 |
| 5 | 1 |

TABLE II
VALUE OF $\beta_p$ FOR DIFFERENT INPUT FEATURES FOR HANG

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $\beta_p$ | 2.53 | 2.54 | 0.00 | 0.00 |
| $1-e^{-\beta_p^2}$ | 0.99 | 0.99 | 0.00 | 0.00 |

defined on the input and output variables which could improve the results further.

The values of $\beta_p$ for the various features and the corresponding values of $1-e^{-\beta_p^2}$ after the Phase 1 training are given in Table II. Table II clearly shows that the network is able to indicate features $x_3$ and $x_4$ as not important and eliminate their effect completely on the output. In this case, as Table I shows, we started with 256 nodes in layer 3, i.e., 256 antecedent clauses. Also, as layer 4 contains 5 nodes, the initial architecture represented $256 \times 5 = 1280$ rules. But Phase 1 of training indicates that two features are redundant/bad. Before Phase 2 training the network is pruned of the redundant nodes, which reduces the antecedent clauses to 16, hence, the number of rules gets reduced to $16 \times 5$, i.e., 80. Since after Phase 2 incompatible rules are removed, the total number of rules represented by the final architecture is 16. Thus, here we obtain a 99.75% reduction in the number of rules in the final architecture.

We also investigated the generalizing capability of the network. A mesh of 256 points in the range $0 \leq x_1, x_2 \leq 5$ was considered. The network then results in a SSE of 17.07 and a PI of 0.008. The mean square error on the test set was 0.06. The MD of the desired output from the obtained output was 0.79. This proves that the network also has good generalizing capabilities. The difference of the correct surface and the surface produced by our system is shown in Fig. 7.

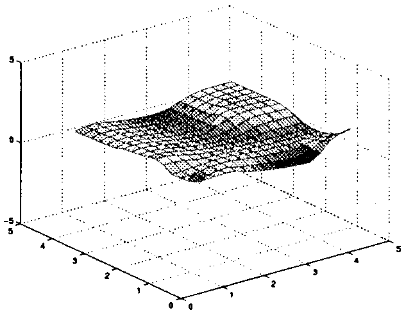

Fig. 7. Difference surface for HANG.

TABLE III
NO OF FUZZY SETS FOR DIFFERENT FEATURES USED WITH CHEM

| Features | No of Fuzzy Sets |
|---|---|
| $x_1$ | 4 |
| $x_2$ | 2 |
| $x_3$ | 4 |
| $x_4$ | 2 |
| $x_5$ | 2 |
| $y$ | 7 |

TABLE IV
INITIAL ARCHITECTURE OF THE NEURAL FUZZY SYSTEM USED FOR CHEM

| layer no. | no. of nodes |
|---|---|
| 1 | 5 |
| 2 | 14 |
| 3 | 128 |
| 4 | 7 |
| 5 | 1 |

## B. Results on CHEM

As described before, this data set has 5 input features, namely, $u_1$, $u_2$, $u_3$, $u_4$, and $u_5$ and a single output feature $y$. The number of input and output fuzzy sets considered are shown in Table III, and the initial number of nodes in the different layers are depicted in Table IV. The membership functions of the various fuzzy sets used for this data set are depicted in Figs. 8–12

For CHEM the learning parameters were $\eta = 0.0001$ and $\mu = 0.000\,01$ and the training was continued for 1000 epochs in Phase 1, 500 epochs in Phase 2, and 3500 epochs in Phase 3. The SSE was reduced from 4 382 539 to 16 231. The PI was equal to 0.0021 after Phase 3. Lin and Cunningham [27] obtained a PI of 0.0022. Sugeno and Yasukawa [36] does not provide any performance measure of their system on this data. The performance of our system is compared with that of the real output in Fig. 13, which exhibits a good match.

The values of $\beta_p$ and $1-e^{-\beta_p^2}$ for the various features after the Phase 1 training are given in Table V. Table V again establishes

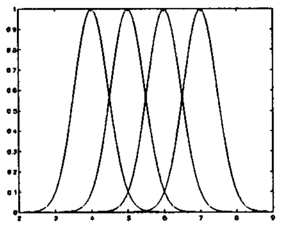

Fig. 8. Membership functions used for $u_1$.

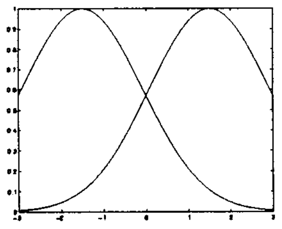

Fig. 9. Membership functions used for $u_2$.

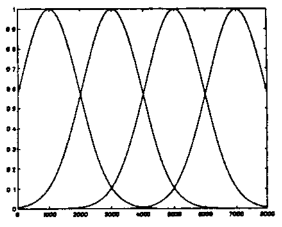

Fig. 10. Membership functions used for $u_3$.

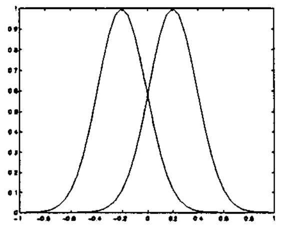

Fig. 11. Membership functions used for $u_4$ and $u_5$.

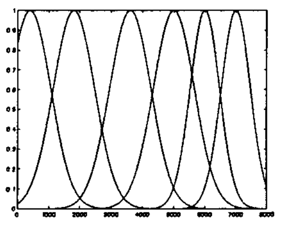

Fig. 12. Membership functions used for $y$.

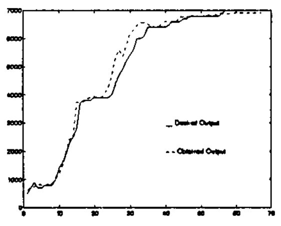

Fig. 13. Performance comparison of the proposed system.

TABLE V
VALUES OF $\beta_p$ FOR DIFFERENT INPUT FEATURES

| | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ |
|---|---|---|---|---|---|
| $\beta_p$ | -2.53 | 1.98 | 1.39 | 0.21 | 0.20 |
| $1 - e^{-\beta_p^2}$ | 0.99 | 0.98 | 0.85 | 0.04 | 0.04 |

the capability of the proposed system in identifying the features that are not important. It clearly shows that $u_4$ and $u_5$ do not contribute significantly to the output of the system—thus they are indifferent or bad features. This result conforms to the findings of Sugeno and Yasukawa [36], who also found that features $u_1$ to $u_3$ are the only important ones. In this case, the number of antecedent clauses at the beginning of Phase 1 training was 128 (Table IV) and the number of nodes in layer 4 was 7. Thus the initial architecture represented $128 \times 7 = 896$ rules. At the end of Phase 1 two features were identified as not important, and thus, pruning of redundant nodes yielded 32 antecedent clauses resulting in $32 \times 7 = 224$ rules. After pruning of the incompatible rules, the final architecture represents 32 rules. Therefore, in this case we obtain 96.42% reduction in the number of rules.

The generalization ability of the network for this data could not be measured as we could not get any data to do so.

## VII. CONCLUSION

A novel scheme for simultaneous feature selection and SI in a neuro-fuzzy framework has been proposed. It is a five-layer network, which can realize a fuzzy rule-based inferencing system and at the same time can find out the features which are not important. We also proposed methodologies for pruning the redundant nodes and incompatible rules that can result in a more readable network. The proposed system has been implemented on several data sets and the results found are quite good.

There are a few issues that have not been considered in the present work. They are as follows. We have not given any guidelines to decide on the number of input and output fuzzy sets and their definitions, which are important for designing a good system. We did not tune the parameters of different membership functions used. Tuning of the membership functions is expected improve the performance further. Our pruning strategy removes redundant nodes and eliminates incompatible rules but still the system considers all possible antecedent clauses, which may not always be required.

The main thrust of this paper was to demonstrate the effectiveness of the proposed network for simultaneous feature analysis and systems identification and it is found to do a good job as revealed by the simulation results.

## REFERENCES

[1] J. Basak, N. R. Pal, and S. K. Pal, “A connectionist system for learning and recognition of structures: Application to handwritten characters,” *Neural Networks*, vol. 8, no. 4, pp. 643–657, 1995.
[2] R. De, N. R. Pal, and S. K. Pal, “Feature analysis: Neural network and fuzzy set theoretic approaches,” *Pattern Recognit.*, vol. 30, no. 10, pp. 1579–1590, 1997.
[3] M. Figureiredo and F. Gomide, “Design of fuzzy systems using neuro-fuzzy networks,” *IEEE Trans. Neural Networks*, vol. 10, pp. 815–827, July 1999.
[4] Y. Hayashi, J. Buckley, and E. Czogala, “Fuzzy neural network with fuzzy signals and weights,” *Int. J. Intell. Syst.*, vol. 8, pp. 527–537, 1993.

[5] S. Haykin, *Neural Networks-A Comprehensive Foundation*. New York: Proc. Con., Inc., 1994.
[6] H. Ishibuchi, R. Fuijoka, and H. Tanaka, "Neural networks that learn from fuzzy if-then rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 85–97, 1993.
[7] N. K. Kasabov, R. I. Kilgour, and S. J. Sinclair, "From hybrid adjustable neuro-fuzzy systems to adaptive connectionist based systems for phoneme and word recognition," *Fuzzy Sets Syst.*, vol. 103, pp. 349–367, 1999.
[8] 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.
[9] J. Keller, R. Krishnapuram, Z. Chen, and O. Nasraoui, "Fuzzy additive hybrid operators for network based decision making," *Int. J. Intell. Syst.*, vol. 9, no. 11, pp. 1001–1024, 1994.
[10] J. Keller and H. Tahani, "Implementation of conjunctive and disjunctive fuzzy logic rules in neural networks," *Int. J. Approx. Reas.*, vol. 6, pp. 221–240, 1992.
[11] J. Keller, R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic," *Fuzzy Sets Syst.*, vol. 45, pp. 1–12, 1992.
[12] J. Kim and N. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, pp. 1301–1319, 1999.
[13] M. A. Khabou, P. D. Gader, and H. Shi, "Entropy optimized morphological shared weight neural networks," *Opt. Eng.*, vol. 38, no. 2, pp. 263–273, 1999.
[14] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic -Theory and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[15] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Velag, 1998.
[16] R. Krishnapuram and J. Lee, "Propagation of uncertainty in neural networks," in *Proc. SPIE Conf. Robot. Comput. Vis.*, Bellingham, WA, 1988, pp. 377–383.
[17] ——, "Propagation of uncertainty in neural networks," in *Proc. SPIE Conf. Robot. Comput. Vis.*, Bellingham, WA, 1989, pp. 592–597.
[18] ——, "Fuzzy connective based hierarchical aggregation networks for decision making," *Fuzzy Sets Syst.*, vol. 46, no. 1, pp. 11–27, 1992.
[19] ——, "Fuzzy-sets-based hierarchical aggregation networks for information fusion in decision making," *Neural Networks*, vol. 5, pp. 335–350, 1992.
[20] K. Lee, D. Kwang, and H. L. Wang, "A Fuzzy neural network model for fuzzy inference and rule tuning," *Int. J. Uncertainty, Fuzz., Knowl.-Based Syst.*, vol. 2, no. 3, pp. 265–277, 1994.
[21] S. C. Lee and E. T. Lee, "Fuzzy neural networks," *Math. Bio. Sci.*, vol. 23, pp. 151–177, 1975.
[22] C. C. Li and C. J. Wu, "Generating fuzzy rules for a neural fuzzy classifier," in *Proc. 3rd IEEE Int. Conf. Fuzzy Systems*, Orlando, 1994, pp. 1719–1724.
[23] C. T. Lin and I. F. Chung, "A reinforcement neuro-fuzzy combiner for multiobjective control," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 726–744, Dec. 1999.
[24] C. T. Lin and C. S. G. Lee, "Neural network based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, pp. 1320–1335, Dec. 1993.
[25] ——, *Neural Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[26] C. T. Lin and Y. C. Lu, "A neural fuzzy system with linguistic teaching signals," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, pp. 169–189, 1995.
[27] Y. Lin and G. A. Cunningham, III, "A new approach to fuzzy neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, pp. 190–198, 1995.
[28] N. R. Pal, "Soft computing for feature analysis," *Fuzzy Sets Syst.*, vol. 103, pp. 201–221, 1999.
[29] N. R. Pal and K. K. Chintalapudi, "A connectionist system for feature selection," *Neural, Parallel, Sci. Comput.*, vol. 5, no. 3, pp. 359–381, 1997.
[30] 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*, Piscataway, NJ, 1997, pp. 2524–2529.
[31] 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.
[32] N. R. Pal and T. Pal, "On rule pruning using fuzzy neural networks," *Fuzzy Sets Syst.*, vol. 106, pp. 335–347, 1999.
[33] S. K. Pal and N. R. Pal, "Soft computing: Goals, tools, and feasibility," *J. Inst. Elect. Telecommun. Eng.*, vol. 42, no. 4-5, pp. 195–204, 1996.
[34] R. Marco, "Comments on a new approach to fuzzy-neural systems modeling," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 2, pp. 209–210, 1996.
[35] J. J. Shann and H. C. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems," *Fuzzy Sets Syst.*, vol. 71, pp. 345–357, 1995.
[36] M. Sugeno and T. Yasukawa, "A Fuzzy-Logic based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 7–31, 1993.
[37] Y. Wong, P. D. Gader, and P. C. Coffield, "Morphological shared weight neural networks with applications to automatic target detection," *IEEE Trans. Neural Networks*, vol. 8, pp. 1195–1203, Sept. 1997.
[38] S. Wu and M. J. Er, "Dynamic fuzzy neural networks—a novel approach to function approximation," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 358–363, Apr. 2000.
[39] S. Yao, C. Wei, and Z. He, "Evolving fuzzy neural networks for extracting rules," in *Proc. 5th IEEE Int. Conf. Fuzzy Syst.*, New Orleans, LA, 1996, pp. 361–367.

**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 the Indian Statistical Institute, Calcutta, in 1999. He is currently pursuing the Ph.D. degree in 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'00) received the B.Sc. degree with honors in physics and the Masters of Business Management degree from the University of Calcutta, India, in 1979 and 1982, respectively, and the M.Tech. and Ph.D. degrees in computer science from the Indian Statistical Institute, Calcutta, in 1984 and 1991, respectively.

Currently, he is a Professor in the Electronics and Communication Sciences Unit of the Indian Statistical Institute. From September 1991 to February 1993, July 1994 to December 1994, October 1996 to December 1996, and January 2000 to July 2000, he was with the Computer Science Department of the University of West Florida, Pensacola. He was also a Guest Faculty of the University of Calcutta. His research interests include image processing, pattern recognition, fuzzy sets theory, measures of uncertainty, neural networks, genetic algorithms, and fuzzy logic controllers. He coauthored the book *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Norwell, MA: Kluwer, 1999) and coedited a volume in *Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99)*. He is an Associate Editor of the *International Journal of Fuzzy Systems* and the *International Journal of Approximate Reasoning*,

Dr. Pal is an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS, and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS (electronic version).