The Third AFSS(1998), 725－730

Extraction of Fuzzy Rules with Importance for Classifier Design

Kuhu Pal

Machine Intelligence Unit, Indian Statistical Institute
203 B. T. Road, Calcutta - 700 035, INDIA
email: res9517@isical.ernet.in

Abstract : Recently we extended the fuzzy model for rule based systems incorporating an importance factor for each rule. The model permits for both unrestricted as well as non-negative importance factors. We use this extended model to design a fuzzy rule based classifier system which uses both the firing strength of the rule and the importance factor to decide the class label. The effectiveness of the scheme is established using several data sets.

1 Introduction

The problem of extracting fuzzy rules for pattern classification is a difficult task. In pattern classification a new pattern is classified using some system designed with a collection of correctly classified samples, called training set. Generally in a rule based classifier [1] each rule is given equal importance. But this may not be desirable. All rules may not have the same level of discriminating power. Rules obtained through exploratory data analysis may not always be very reliable. Moreover, rules may be correct, but all rules may not have equal importance. Rule tuning is usually achieved through modification of membership functions. Effect of changing a membership function is global in the sense, it influences all rules that involve that particular membership function. We cannot adjust only one rule in isolation.

This problem is applicable not only to classifier systems, but for all fuzzy rule based systems including control systems. To overcome these problems and to incorporate relative importance of rules in the fuzzy model, we proposed an extension of the conventional fuzzy model [2-4].

Here we use this extended model with suitable modification of the error function for designing a rule based classifier. Exploratory data analysis [1,5] is used to extract the initial rule base which is then further tuned. The non-negative importance factors can also be used for rule selection. We have tested our model on several data sets and obtained encouraging results.

The rest of the paper is organized as follows. Section 2 discusses the extended fuzzy model. Some rule extraction schemes have been discussed in Section 3. Use of exploratory data analysis for rule extraction and the subtractive clustering method are presented in Section 4; while the proposed scheme is included in Section 5. Our results are presented in Section 6, while the paper is concluded in Section 7.

2 The Extended Model

A fuzzy system is defined in terms of if-then rules. Given some training data, a set of fuzzy if-then rules describing the relation between the input and output can be extracted in many ways, but here we restrict ourselves only to methods aided by clustering. Each rule has two parts - the antecedent part and the consequent part. The antecedent part gives the condition of the system and the consequent part tells the action to be taken. Depending on the form of the consequent there are two types of models: the *Mamdani-assilian* (MA) model [6] and the *Takagi-Sugeno* (TS) model [7]. Consider a p-input $(x_1, x_2, \cdots, x_p)^T \in \Re^p$ and 1-output $y \in \Re$ system. Then under MA model, a rule, say the r^{th} rule, takes the form

R(r) : If (x_1 *is* X_{r1}) and ... (x_p *is* X_{rp}) then $u = U_r$; r=1,2,...,k.

Here X_{rj}'s are fuzzy sets defined on x_r and U_r's are fuzzy sets on y.

On the other hand, for the TS model each rule provides a crisp action. The consequent part is defined using a function giving a crisp value. The r^{th} rule, R(r), takes the form

R(r) : If (x_1 *is* X_{r1}) and ... (x_p *is* X_{rp}) then $u = u_r = F_r(x_1, .., x_p)$; $r = 1, 2, ..., k$;

where X_{rj}'s are fuzzy sets defined on x_r and u_r's are crisp values provided by the function F_r.

Since our investigation is related to the TS model we describe it in brief. For given values of input variables, $x_1, x_2, .., x_p$, the membership value $\mu(x_i)$ is calculated for each i = 1,..,n. Each $\mu(x_i)$ gives the extent to which the corresponding fuzzy set is satisfied.

The minimum or the product of all $\mu(x_i)$s is usually taken as the firing strength. We denote the firing strength of the r^{th} rule by f_r. The firing strength modulates the output (consequent) value. Finally the system output may be computed as the weighted normalized sum of all pairs (f_r, u_r) :

$$y_i' = \frac{\sum_{r=1}^{k} f_r * u_r}{\sum_{r=1}^{k} f_r} \quad (1)$$

2.1 Extended Model with Rule Importance

Deciding on the rules to be used by a fuzzy system is a difficult task. Even when experts are available, it is often difficult to extract the correct rules from them. Existence of just a single incorrect or inconsistent rule may degrade the performance of the system significantly. For the sake of arguments let us assume that experts provided rules are correct and consistent. Normally each such rule is given equal importance. But, for all systems this may not be desirable. Different rules may have different importance. Moreover, tuning of a fuzzy set, say A, influences all rules that involve A. Thus, alteration of a fuzzy set has a global impact on the rule-base. The designer usually cannot adjust only one rule in isolation. To overcome these problems we incorporate relative importance to each rule in the fuzzy model [2-4]. Here under the TS framework r^{th} rule takes the form

R(r) :If $(x_1\ is\ X_{r1})$ and...$(x_p\ is\ X_{rp})$ then $u = u_r = F_r(x_1, .., x_p)$ with importance α_r,

α_r is unrestricted in sign. Now for a given input $\mathbf{x} \in \Re^p$, the defuzzified value may be computed as

$$y_i' = \frac{\sum_{r=1}^{k} f_r * u_r * \alpha_r}{\sum_{r=1}^{k} f_r * \alpha_r} \quad (2)$$

where f_r is the firing strength of the r^{th} rule and k is the total number of rules that are fired. This model can be used to deal with inconsistent rules provided we have adequate training data. Given a training data (X,Y), $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subseteq \Re^p$ and $Y = \{y_1, y_2, ..., y_n\} \subseteq \Re$, we can obtain a suitable set of values for α_r minimizing $\sum_{i=1}^{n} (y_i - y_i')^2$. Note that theoretically, the denominator $\sum_{r=1}^{k} f_r * \alpha_r$ could be very small, even zero. But the training process will never allow this. Because, if $\sum_{r=1}^{k} f_r * \alpha_r \rightarrow 0$ then $y_i' \rightarrow \infty$, but y_i is finite. This will result in infinite error. Thus when α_r's are learnt using training data with finite output values, $\sum_{r=1}^{k} f_r * \alpha_r$ can never be zero. In other words, if we start with $\alpha_r = 1\ \forall$ r, then the gradient based tuning algorithm (or any consistent tuning algorithm) will never change α in such a manner that $\sum_{r=1}^{k} f_r * \alpha_r$ goes to zero. And if the input-output relation is smooth and the training data that are used to learn the α_r, adequately represent the relation so that the actual input-output relation is captured by the identified fuzzy system, then $\sum_{r=1}^{k} f_r * \alpha_r \approx 0$ is not expected to occur for any future data - that follow the characteristics of the training data. However, if we could restrict α_r to be *non-negative* ($\alpha \geq 0$), then it would enable us to tune the rule-base and at the same time it can solve the difficult problem of *rule selection* easily. And obviously, the chance of the degenerate case of $\sum_{r=1}^{k} f_r * \alpha_r \approx 0$ is eliminated.

This paper is about non-negative importance. In order to impose the non-negativity constraint, we modify the ETS model with rules of the form :

R(r) :If $(x_1\ is\ X_{r1})$ and...$(x_p\ is\ X_{rp})$ then $u = u_r = F_r(x_1, .., x_p)$ with importance $\beta_r = {\alpha_r}^2$

where α_r is *unrestricted* in sign, but the actual importance ($\beta_r = {\alpha_r}^2$) is always positive. The associated defuzzification scheme is

$$y_i' = \frac{\sum_{r=1}^{k} f_r * u_r * (\alpha_r)^2}{\sum_{r=1}^{k} f_r * (\alpha_r)^2} \quad (3)$$

Using gradient descent, the update equations for α_r can be derived as

$$\alpha_r(t+1) = \alpha_r(t) - \eta_\alpha(t) * \frac{(y_i' - y_i) * (u_r - y_i')}{\sum_{j=1}^{k} f_j * {\alpha_j}^2}, \quad (4)$$

where η_α is the learning co-efficient for α.

We update the importance factor using (4), with each pair of $(\mathbf{x}_i, y_i)$. The process is repeated until $||\ \alpha(t)\text{-}\alpha(t+1)||\ /k < \epsilon$, where ϵ is a prefixed small positive quantity. The learnt value of α_r could be negative, but $\beta_r = {\alpha_r}^2$ would always be non-negative.

3 Some Rule Extraction Schemes

Here we consider the problem of extracting fuzzy *if-then* rules for describing the input-output behavior of a system directly from numerical data using clustering. Let us denote the set of p-dimensional input vectors as $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \Re^p$, and the associated set of q-dimensional output vectors as $Y = \{\mathbf{y}_1, ..., \mathbf{y}_n\} \subset \Re^q$. Clustering an unlabeled data set $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \Re^p$ is the partitioning of X and hence, to the object generating X, into c, $1 < c < n$ subgroups such that each subgroup represents a *natural* substructure present in X. In other words, clustering finds *homogeneous* groups in X. The

definition of *natural* substructures or *homogeneous* groups often depends on the problem at hand. Formally, clustering can be described as the assignment of labels to the vectors in X, and hence, to the objects generating X so that similar objects get similar labels. If the labels are hard (crisp), we hope they identify c natural subgroups in X. c-partitions of X are sets of (cn) values $\{u_{ik}\}$ that can be conveniently arrayed as a $c \times n$ matrix $U = [u_{ik}]$.

We now discuss how clustering results can be used to extract rules. We define

$$X^* = \left\{ \mathbf{x}_i^* = \begin{pmatrix} \mathbf{x}_i \in \mathcal{R}^p \\ \mathbf{y}_i \in \mathcal{R}^q \end{pmatrix} \in \mathcal{R}^{p+q}, i = 1, ..., n \right\}$$

i.e., $\mathbf{x}_i^*$ is nothing but $\mathbf{x}_i$ augmented by $\mathbf{y}_i$. We cluster X^* by some clustering algorithm producing a set of centroids

$$V^* = \left\{ \mathbf{v}_i^* = \begin{pmatrix} \mathbf{v}_i^x \in \mathcal{R}^p \\ \mathbf{v}_i^y \in \mathcal{R}^q \end{pmatrix} \in \mathcal{R}^{p+q}, i = 1, ..., c \right\}$$

and a partition matrix. This clustering result can be used to extract fuzzy rules [1,5,11]. Use of clustering results for fuzzy rule extraction is motivated by the fact that if there is a cluster in the input space with centroid $\mathbf{v}_i^x$ and we assume a smooth relationship between the input and output, then the points in the output space corresponding to the input cluster are likely to form a cluster around $\mathbf{v}_i^y$. And this local input-output relation can be represented by an if-then fuzzy rule. On the other hand, when $\mathbf{v}_i^*$ is associated with a good cluster in the input-output space, then this is a signal that when $||\mathbf{x}_k - \mathbf{v}_i^x||$ is small, $||\mathbf{y}_k - \mathbf{v}_i^y||$ would also be small. This is again a rough indication that such a cluster represents a locally continuous or even smooth input-output relation.

In case of classifier design, the output vectors are class label vectors. Hence clustering of the input-output data may not be meaningful. The output data have as many natural clusters as the number of classes. Thus the input data set corresponding to each class is clustered to extract rules for that class.

Chung and Lee [8] used a variant of Kohonen's learning vector quantization algorithm to learn the center of the membership function. While Sun and Jang learned the membership function centers to optimize membership functions as well as parametrized fuzzy operators [9]. Yager and Filev used the mountain method (MCM) to cluster X^* [10,11]. They clustered in the input-output space for rule generation. The optimal number of clusters is chosen based on a user defined threshold δ on the mountain potential. Consider a MISO system with input $\mathbf{x} \in \Re^p$ and output $y \in \Re$. A cluster centroid is then converted into a fuzzy rule of the form:

If x is CLOSE to $v_i{}^x$ then y is CLOSE to $v_i{}^y$.

Now we write

A_i = CLOSE to $v_i{}^x$ and B_i = CLOSE to $v_i{}^y$,

and we get a set of c rules:

If x is A_i then y is B_i; i = 1, ...,c.

Each antecedent clause If x is A_i is then translated into p atomic clauses: x_k is A_{ik}; k = 1, ...,p. They used Gaussian type membership functions to model A_{ij} and B_i:

$$A_{ij}(x_j) = exp(\tfrac{1}{2\sigma_{ij}{}^2} \ (x_j - v_{ij}{}^x)^2) \text{ and}$$

$$B_i(x) = exp(\tfrac{1}{2\sigma^2} \ (y - v_i{}^y)^2).$$

σ_{ij} is the spread of the j^{th} antecedent of the i^{th} rule and σ is the spread of the consequents. Yager and Filev used the height method of defuzzification with the MA model. (This can equivalently be viewed as the TS model with zero order function for the consequents). The initial estimates of the parameters σ_{ij} are taken as $\sqrt{(\frac{1}{2\beta})}$, where β is a parameter used in the mountain function for clustering. All parameters of the system ($v_{ij}{}^x$, $v_i{}^y$, σ_{ij}) are then further tuned with gradient descent to minimize the total square error. Although, MCM determines the number of clusters automatically, it is strongly influenced by the parameters of the mountain potential function and the threshold value used to stop the clustering process. An inappropriate choice of these parameters may over-determine or under-determine the number of rules.

Chiu [1] presented a modified method of MCM known as subtractive clustering method for extracting fuzzy rules for pattern classification. The initial membership functions for the rules were obtained from cluster regions. He also used gradient descent to minimize a classification error measure.

We can use any cluster estimation method for clustering the training data of each class. In the present case the initial rule extraction is done using the *Subtracting Clustering* method of Chiu [1] and hence we discuss it in details next.

4 The Subtractive Clustering

Chiu [1] used a modified form of MCM, the subtractive clustering method (SCM), to extract rules from numerical data for pattern classification. For this system, outputs are class labels. The input data set X is first partitioned into c subsets

$X = \bigcup_{i=1}^{c} X_i, X_i \cap X_j = \phi,\ i \neq j$ according to actual class labels, i.e., all $\mathbf{x} \in X_i$ have the same output value. Then each X_i is clustered using SCM. Suppose $v_{ij} \in \Re^p$ is the centroid of the j^{th} cluster in X_i. Then this cluster is transformed into a rule of the form

If x is CLOSE to v_{ij} then y is i (i.e., x is in class i), $\forall\ j = 1, ..., c_i$, i = 1,...,c.

Here also, x is CLOSE to v_{ij} is actually translated into p atomic antecedent clauses. Each fuzzy set CLOSE to is modeled by a Gaussian membership function. Like Yagar and Filev, Chiu also fine tuned the initial model thus obtained using gradient descent but on a different type of error function defined by

$$E = \frac{1}{2}(1 - \mu_{c,max} + \mu_{\neg c,max})^2 \qquad (5)$$

where $\mu_{c,max}$ is the maximum degree of fulfillment among all rules that infer the correct class c and $\mu_{\neg c,max}$ denotes the maximum firing strength of all rules that do not infer the class c.

Like MCM, SCM also does not use any cluster validity index and the number of clusters SCM settles to is dependent on the parameters of the mountain function. In order to validate and optimize the system, Chiu used different orders of the TS model (for learning input-output relation) and also different values for the parameters of the mountain function (hence different number of rules). The system evaluation is done in terms of RMS modeling error.

For a group of n_i training data $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n_i}\} \subset \Re^p$ representing a particular class, they considered each data point as a potential cluster center and defined a measure of the potential of data point x_i as

$$P_i = \sum_{j=1}^{n} exp(-\alpha || x_i - x_j ||^2),$$
$$\alpha = 4 \ / \ {r_a}^2,$$

where r_a is a positive constant, defining the radius of a neighborhood; data points outside this radius have little influence on the potential. || .. || is an Euclidian distance. After calculating the potential of all data points, the highest potential is selected as the first cluster center.

The potential of each data point is then revised by

$$P_i \Longleftarrow P_i - {P_1}^* exp(-\beta\ ||\ x_i - {x_1}^*\ ||^2)$$

where ${P_1}^*$ and ${x_1}^*$ are the potential value and location of the first cluster, $\beta = 4/{r_b}^2$, r_b is a positive constant. Typically $r_b = 1.25 * r_a$. After revision, the data point with the highest remaining potential is selected as the second cluster center. For k^{th} such operations, the potential of each data point was revised by

$$P_i \Longleftarrow P_i - {P_k}^* exp(-\beta\ ||\ x_i - {x_k}^*\ ||^2) \qquad (6)$$

where, ${P_k}^*$ and ${x_k}^*$ are the potential and location of the k^{th} cluster. The process was repeated until ${P_k}^* < 0.15\ {P_1}^*$.

If the i^{th} cluster center ${x_i}^*$ is found in the group of data for class 1, the corresponding rule can be written as

R(i):If {x is CLOSE to ${x_i}^*$} then class is c1.

The fuzzy set {x is CLOSE to ${x_i}^*$} is modeled by

$$\mu_i = exp(-\alpha||\ x - {x_i}^*\ ||^2) \qquad (7)$$

The rule R(i) can be written in a more familiar form:

If X_1 is A_{i1} & X_2 is A_{i2} & $\cdots$ then class is c1.

where X_j is the j^{th} input feature and A_{ij} is the membership function in the i^{th} rule associated with the j^{th} input feature. The membership function A_{ij} is given by

$$A_{ij}X(j) = exp\{-\frac{1}{2}(\frac{X_j - {x_{ij}}^*}{\sigma_{ij}})^2\} \qquad (8)$$

where ${x_{ij}}^*$ is the j^{th} element of ${x_i}^*$, and ${\sigma_{ij}}^2 = \frac{1}{(2\alpha)}$

After obtaining the initial rule set, gradient descent algorithm is used to tune ${x_{ij}}^*$ and σ_{ij} to minimize a classification error measure. The update equations for centroids and σ following gradient descent are:

$${x_{ij}}^* \leftarrow {x_{ij}}^* - \lambda \frac{\partial E}{\partial {x_{ij}}^*}$$

and

$$\sigma_{ij} \leftarrow \sigma_{ij} - \lambda \frac{\partial E}{\partial \sigma_{ij}}$$

where λ is the learning co-efficient. After simplification, these equations can be written as

$${x_{ij}}^* \Longleftarrow {x_{ij}}^* \pm \lambda \mu_i \frac{(1 - \mu_{c,max} + \mu_{\neg c,max})(X_j - {x_{ij}}^*)}{{\sigma_{ij}}^2} \qquad (9)$$

and

$${\sigma_{ij}}^* \Longleftarrow {\sigma_{ij}}^* \pm \lambda \mu_i \frac{(1 - \mu_{c,max} + \mu_{\neg c,max})(X_j - {x_{ij}}^*)}{{\sigma_{ij}}^3} \qquad (10)$$

These update formulae are applied to only two rules: the rule that provides good result ($\mu_{c,max}$) and the rule that provides bad result ($\mu_{\neg c,max}$). Here X_j is the value of the j^{th} input feature of the data sample and μ_i is the degree of fulfillment of rule i.

5 Proposed Classifier with Rule Importance

In this section we redefine the error function, derive the learning equations for relative importance factor $\beta_r = (\alpha_r{}^2)$, and describe tuning algorithms for them.

Suppose the input data are in p-dimension; i.e., each $\mathbf{x} \in R^p$; $\mathbf{x} = (x_1, x_2, ..., x_p)^T$, then the proposed system consists of rules R(i), of the form:

If X_1 is A_{i1} & X_2 is A_{i2} & $\cdots$ then class is c_i with importance β_i

To make β_i non-negative, we take $\beta_i = \alpha_i{}^2$.

Here X_j and A_{ij} have the same meaning as earlier. In fact, the membership function A_{ij} is the same as equation (8).

After getting the initial rules using subtractive clustering method, we use a gradient descent algorithm to tune the importance factor to minimize a new classification error measure. For a data point belonging to a class c, we use the error measure

$$E = \frac{1}{2}(1 - \beta_{c,max}\mu_{c,max} + \beta_{\neg c,max}\mu_{\neg c,max})^2$$

$$= \frac{1}{2}(1 - \alpha_{c,max}{}^2\mu_{c,max} + \alpha_{\neg c,max}{}^2\mu_{\neg c,max})^2 \quad (11)$$

where $\beta_{c,max}$ is the importance factor of the rule with the highest degree of fulfillment among all rules that infer class c and $\beta_{\neg c,max}$ is the importance factor of the rule with the highest degree of fulfillment among all rules that do not infer class c.

The importance factors β_rs are updated through α_rs using the following gradient descent formula:

$$\alpha_{c,max} \Longleftarrow \alpha_{c,max} - \lambda_\alpha \frac{\partial E}{\partial \alpha_{c,max}}$$

and

$$\alpha_{\neg c,max} \Longleftarrow \alpha_{\neg c,max} - \lambda_\alpha \frac{\partial E}{\partial \alpha_{\neg c,max}}$$

where λ_α is the learning co-efficient. After simplification, these equations can be written as

$$\alpha_{c,max} \Longleftarrow \alpha_{c,max} + \lambda_\alpha * (1 - \alpha_{c,max}{}^2\mu_{c,max} + \alpha_{\neg c,max}{}^2\mu_{\neg c,max}) * (\alpha_{c,max}\mu_{c,max}) \quad (12)$$

and

$$\alpha_{\neg c,max} \Longleftarrow \alpha_{\neg c,max} - \lambda_\alpha * (1 - \alpha_{c,max}{}^2\mu_{c,max} + \alpha_{\neg c,max}{}^2\mu_{\neg c,max}) * (\alpha_{\neg c,max}\mu_{\neg c,max}) \quad (13)$$

Here also these update formulae (12) and (13) are applied to only two rules: the rule that provides good result ($\alpha_{c,max}$) and the rule that support bad result ($\alpha_{\neg c,max}$).

The algorithm for tuning α_r proceeds as follows: For each data point update α_r using (12) and (13). Repeat the process until $\| \alpha(t) - \alpha(t+1) \| / k < \epsilon$, where ϵ is a small positive quantity and $\alpha(t)$ is the vector of weights after t epochs. A complete pass through the data is called an epoch.

6 Result

6.1 An Application in Vowel Recognition

In order to illustrate the effectiveness of the proposed technique we use it to develop several pattern recognition applications, including a vowel recognition system for Telugu vowels [12]. We report here the results for a synthetic data set containing 600 points in 4 dimension. The data set is generated from four 4-variate normal distributions. From each class 150 points are drawn. There are substantial overlap between the four classes. We randomly divided the data into two subsets X_{Tr} and X_{Ts} such that $X_{Tr} \bigcap X_{Ts} = \phi$, $X_{Tr} \bigcup X_{Ts} = X$, $| X_{Tr} | = 200$ and $| X_{Ts} | = 600$. We use X_{Tr} to extract the initial rule-base.

In this investigation we use $r_a = 1.0$. We obtain 14 rules: 4 rules for class 1, 5 rules for class 2, 2 rules for class 3 and 3 rules for class 4. The initial importance of each rule is 1. After tuning for the centroids and σ using (9) and (10), the square error on X_{Tr} reduced from 176.06 to 47.63. We then tuned β_r for this rule-base, and this resulted in a total square error of 32.18 (i.e., a reduction of 30% approximately). For this data set, weight tuning is found to increase the number of misclassification by 1. But as we shall see now, it enhances the generalization capability of the rule-base. To see this we tested with the rule-base (both after membership tuning and after membership tuning followed by weight tuning) on the test data X_{Ts}. The Confusion Table for X_{Ts} before and

after weight tuning are shown in Table I and Table II, respectively. It is interesting to note that the total square error on X_{Ts} for the rule-base tuned for only membership is 122.76 and the misclassification (M) in this case is 15. On the other hand, the total square error on X_{Ts} with the rule-base tuned for both membership and weight is reduced to 87.14 (about 28% reduction). The misclassification (M) is also found to be reduced to 14. Note that the correct class is determined by taking the maximum of $(\mu_c.\beta_c)$, not using the maximum of μ_c only.

95	2	0	3
0	97	0	3
1	2	96	1
2	0	1	97

Table I
Confusion table
(before β tuning)
M = 15

94	3	0	3
0	98	0	2
1	2	96	1
2	0	0	98

Table II
Confusion table
(after β tuning)
M = 14

7 Conclusion

We proposed a fuzzy rule-based classifier system which can extract rules directly from numerical data. The system assigns an importance factor (a non-negative real value) to each rule. Learning algorithm for obtaining an optimal value of the importance factor for each rule is also derived. The proposed system with importance factor is found to reduce the total square error significantly compared to the system with equal importance for each rule. Further investigation with many real life data sets to be done to establish the effectiveness of the proposed model. The possibility of rule elimination for classifier using the importance factors will also be investigated in future.

Acknowledgments

I gratefully acknowledge the partial support by the Council of Scientific and Industrial Research, India for completion of this work. I like to thank Prof. S. K. Pal, Indian Statistical Institute, Calcutta for his interest in this work.

References

1. S. L. Chiu, **Extracting Fuzzy Rules for Pattern Classification by Cluster Estimation** *Proc. sixth Int. Fuz. Systs. Assoc., World Congress (IFSA '95),* Sao Paulo, Brazil, 1-4, July 1995.

2. K. Pal (nee Dutta) and N. R. Pal, **A Flexible Fuzzy Controller with Relative Importance of Rules**, *Proc. 4 th Int. Conf. of Soft Computing, IIZUKA'96*, pp 398-401, September 30 - October 5 (1996), IIZUKA, Fukuoka, Japan.

3 K. Pal (nee Dutta) and N. R. Pal, **Learning of Rule Importance for Fuzzy Controllers to deal with Inconsistent Rules and for Rule Elimination**, *Journal of Control and Cybernetics*, (Accepted).

4. N. R. Pal and K. Pal (nee Dutta), **Handling of Inconsistent Rules with an Extended Model of Fuzzy Reasoning** , *J. Intelligent & Fuzzy Systems*, (Accepted).

5. N. R. Pal, K. Pal, J. C. Bezdek and T. Runkler, **Some Issues in System Identification using Clustering**, *Int. Joint Conf. on Neural Networks, ICNN 1997*, IEEE Press, Piscataway, NJ, 2524-2529.

6. E. H. Mamdani and S. Assilian, **An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller**, *Int. J. Man Mach. Studies,* vol. 7, no. 1, pp. 1-13, 1975.

7. T. Takagi and M. Sugeno, **Fuzzy Identification of Systems and its Applications to Modeling and Control**, *IEEE Trans. Syst. Man and Cybern.*, Vol. 15, 116-132, 1985.

8. F. L. Chung and Lee, **A Fuzzy Learning Method for Membership Function Estimation and Pattern Classification**, *Proc. 3 rd IEEE Int. Conf. on Fuzzy Systems*, 426-431, Orlando, USA, 1994.

9. C. T. Sun and J. S. Jang, **A Neuro-Fuzzy Classifier and its Applications**, *Proc. 2 nd IEEE Int. Conf. on Fuzzy Systems*, 94-98, San Francisco, USA, 1993.

10. R. R. Yager and D. P. Filev, **Generation of Fuzzy Rules by Mountain Clustering**, *J. Int. & Fuzzy Systems* Vol. 2, 209-219, 1994.

11. R. R. Yager and D. P. Filev, **Approximate Clustering by Mountain Method**, *IEEE Trans. Syst. Man and Cybern.*, Vol. 24, No. 8, 1279-1283, 1994.

12. K. S. Ray and J. Ghosal, **Neuro Fuzzy Approach to Pattern Recognition**, *Neural Networks*, Vol. 10, No. 1, 161-182, 1997.