Fuzzy Logic Approaches to Structure Preserving Dimensionality Reduction

Nikhil R. Pal, *Senior Member, IEEE*, Vijaya Kumar Eluri, and Gautam K. Mandal

***Abstract*—Sammon's nonlinear projection method is computationally prohibitive for large data sets, and it cannot project new data points. We propose a low-cost fuzzy rule-based implementation of Sammon's method for structure preserving dimensionality reduction. This method uses a sample and applies Sammon's method to project it. The input data points are then augmented by the corresponding projected (output) data points. The augmented data set thus obtained is clustered with the fuzzy C-means (FCM) clustering algorithm. Each cluster is then translated into a fuzzy rule to approximate the Sammon's nonlinear projection scheme. We consider both Mamdani–Assilian (MA) and Takagi–Sugeno (TS) models for this. Different schemes of parameter estimation are considered. The proposed schemes are applied on several data sets and are found to be quite effective to project new points, i.e., such systems have good predictability.**

***Index Terms*—Dimensionality reduction, fuzzy rule-based systems, nonlinear projection, Sammon's method.**

I. Introduction

FEATURE extraction and dimensionality reduction are important problems in pattern recognition and exploratory data analysis. Feature analysis can avoid the "curse of dimensionality," improve generalization ability of classifiers by eliminating harmful features and reduce the space and computational requirements associated with analysis of the data. Many features not only lead to more computational overhead, but often can create confusion thereby degrading the performance of a classifier or any other system designed on them. Dimensionality reduction can be done mainly in two ways: selecting a small but important subset of features and generating (extracting) a lower dimensional data preserving the distinguishing characteristics of the original higher dimensional data. This paper deals with extraction of lower dimensional data.

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of n feature vectors (signals) in $\mathcal{R}^p$. The jth observed object (some physical entity) has vector $\mathbf{x}_j$ as it's numerical representation; x_{jk} is the kth characteristic (or feature) associated with object j. Feature extraction and data projection can be viewed as an implicit or explicit mapping Φ from a p-dimensional input space to a q-dimensional output space

$$\Phi: \mathcal{R}^p \rightarrow \mathcal{R}^q \tag{1}$$

such that some criterion, C, is optimized. Usually $q \leq p$, but for some feature extraction problems q may be greater than p also.

A large number of approaches for feature extraction and data projection are available in the pattern recognition literature [1]–[11]. These approaches differ from each other in the characteristics of the mapping function Φ, how Φ is learned, and what optimization criterion C is used. The mapping function can be *linear* or *nonlinear*, and can be learned through *supervised* or *unsupervised* methods.

Sammon's nonlinear projection method [3] is quite effective for small data sets. For large data sets the computational overhead is very high. Moreover, Sammon's method does not have predictability; in other words, with every new data point, the entire process is to be repeated. In order to equip Sammon's method with generalizability, some neural network (NN) implementations have been attempted [10], [11]. NN-based schemes usually work fine but sometimes the performance on the test data becomes poor. More importantly, the NN-based schemes always produce some output even when the test data are far away from the training data or are outliers. For example, even if an input is far away from the convex hull of the training data, the network will produce some lower dimensional representation for it. It would be more useful, if we can develop a system that is interpretable [not a black box like the multilayer perceptron (MLP)], can interpolate (i.e., has predictability) and can deal with outliers. At least the system should be able to reject the inputs which are far away from the training data. Fuzzy rule-based systems have all these three desirable properties, and hence, in this paper we explore the possibility of using fuzzy reasoning system for structure preserving dimensionality reduction. These schemes integrate structure preserving characteristic of Sammon's function and the generalization capability of fuzzy rule based systems. As to the knowledge of the authors so far no attempt has been made to exploit the power of fuzzy rule based system in dimensionality reduction.

The proposed schemes are found to produce, like Sammon's algorithm, lower dimensional data which are coherent with the original data at a lower computational cost. The performance of the proposed schemes have been compared with original Sammon's algorithm and NN implementations of Sammon's method. The results obtained are quite satisfactory.

II. Sammon's Nonlinear Projection Method

Sammon [3] proposed a simple yet very useful nonlinear projection algorithm that attempts to preserve the structure present in a set of n points in p-space by finding n points in q-space such that interpoint distances in q-space approximate the corresponding interpoint distances in p-space.

Let $X = \{\mathbf{x}_k \mid \mathbf{x}_k = (x_{k1}, x_{k2}, \ldots, x_{kp})^T, k = 1, 2, \ldots, n\}$ be the set of n input vectors and let $Y = \{\mathbf{y}_k \mid \mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{kq})^T, k = 1, 2, \ldots, n\}$ be the unknown vectors to be found.

Let $d^*_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{x}_i, \mathbf{x}_j \in X$ and $d_{ij} = d(\mathbf{y}_i, \mathbf{y}_j)$, $\mathbf{y}_i, \mathbf{y}_j \in Y$, where $d(\mathbf{x}_i, \mathbf{x}_j)$ be the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Sammon suggested looking for Y minimizing the error function E

$$E = \frac{1}{\sum\limits_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}}. \tag{2}$$

Sammon used the method of steepest descent for (approximate) minimization of E. Let $\mathbf{y}_i(t)$ be the estimate of $\mathbf{y}_i$ at the tth iteration, $\forall i$. Then, $\mathbf{y}_i(t+1)$ is given by

$$y_{ij}(t+1) = y_{ij}(t) - \alpha \left[\frac{\frac{\partial E(t)}{\partial y_{ij}(t)}}{\left| \frac{\partial^2 E(t)}{\partial y_{ij}(t)^2} \right|} \right], \qquad j = 1, \ldots, q \tag{3}$$

where the nonnegative scalar constant α is the step size for gradient search.

With this method we cannot get an explicit mapping function governing the relationship between patterns in $\mathcal{R}^p$ and corresponding patterns in $\mathcal{R}^q$. Therefore, it is not possible to project new points. Hence, with every additional point, it is necessary to redo the optimization with all data points. Every step within an iteration requires computation of $((n(n-1))/2)$ distances and for large n the computation time becomes high. Finally, there are many local minima on the error surface and it is usually unavoidable for the algorithm to get stuck in some local minimum. When n is large, getting a good solution may be difficult and one may need to try several initializations.

Several modifications of Sammon's algorithm have also been proposed [4]–[7]. These algorithms approximate Sammon's method and do not have predictability. Even for comparison of the quality of outputs, it is desirable to compare with original Sammon's output. Consequently, we do not consider any approximate version of Sammon's algorithm.

III. Artificial Neural Networks for Sammon's Projection

The potential of artificial neural networks (ANNs) in various applications is well established. Recently a number of ANNs have been proposed for feature extraction and multivariate data projection [8]–[11] Next, we discuss an interesting neural implementation of Sammon's method which augments Sammon's algorithm with prediction capability [8], [9].

Let us express Sammon error given in (2) as

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E_{ij}$$

TABLE I
Ten Centroids for IRIS Data Produced by Fuzzy C-Means (FCM)

Cluster	x_1	x_2	x_3	x_4	y_1	y_2
1	0.6116	0.3969	0.1674	0.0132	0.5671	-0.0394
2	0.8874	0.3848	0.7303	0.2539	0.3269	0.5957
3	0.8303	0.3826	0.6440	0.2318	0.3451	0.4941
4	0.8180	0.3719	0.5739	0.1768	0.3906	0.4137
5	0.6307	0.3010	0.3954	0.1223	0.3629	0.1381
6	0.6741	0.4842	0.1774	0.0189	0.6563	0.0352
7	0.6363	0.4236	0.1978	0.0538	0.5710	0.0239
8	0.7233	0.3232	0.5141	0.1406	0.3519	0.2872
9	0.7634	0.3366	0.6275	0.2058	0.2929	0.4172
10	0.6466	0.4394	0.1770	0.0199	0.6066	0.0020

TABLE II
Sorted Centers for x_1 and the Associated Membership Parameters

Cluster	x_1	$b^L_{i,1}$	$b^R_{i,1}$
1	0.6116	0.0379	0.0191
5	0.6307	0.0191	0.0056
7	0.6363	0.0056	0.0093
10	0.6466	0.0093	0.0275
6	0.6741	0.0275	0.0492
8	0.7233	0.0492	0.0401
9	0.7634	0.0401	0.0546
4	0.8180	0.0546	0.0123
3	0.8303	0.0123	0.0571
2	0.8874	0.0571	0.0389

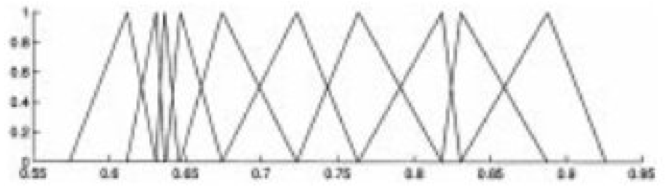

Fig. 1. The ten antecedent membership functions defined on feature x_1 of IRIS.

where

$$E_{ij} = \lambda \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}}$$

with

$$\lambda = \frac{1}{\sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} d^*_{ij}}.$$

d^*_{ij} and d_{ij} are the distances, respectively, in $\mathcal{R}^p$ and $\mathcal{R}^q$ between patterns i and j.

Jain and Mao [8], [9] used a multilayer feedforward network for Sammon's projection. The number of input nodes is set to p. The number of output nodes is equal to q. A pair of input $\mathbf{x}_i \in X, \mathbf{x}_j \in X$ is applied one after another and the corresponding outputs $\mathbf{y}_i$ and $\mathbf{y}_j$ are noted and are used to define E_{ij}. Like an MLP, Jain and Mao trained the net using backpropagation to minimize E.

In [8], it was shown experimentally that the number of nodes required in the hidden layer to be around nq to get good results. Although, it is an interesting application of NN to data projection, it requires a lot of space and training time to get good results. The error after training, as we will see, is also not found comparable to that of Sammon's algorithm.

In [9], another approach was followed for training so as to take advantage of the nonlinearity of the above network. Initially, a PCA network [12], [13] was used to project data and then standard backpropogation algorithm was used to approximate principal components. The connection weights of such a trained MLP were then used to initialize the weights of the Sammon's neural net. This means that Sammon's network is initialized such that it behaves like a PCA network.

The main purpose of this network was to handle nonlinear data,[1] as, linear data are very well projected by the PCA network, but even this may not always be achieved by the proposed implementation [9], as shall be seen from Section V-B.

IV. Proposed Fuzzy Model for Data Projection

Sammon's algorithm does not have predictability, i.e., with every new point the entire data set has to be projected afresh; this in turn reduces the practical utility of Sammon's method. So we want to get a system for Sammon's projection with predictability. Several NN schemes have already been proposed as a solution to the predictability issue, but they have some problems as discussed earlier. Here, we intend to identify the relation between input and the projected data by a set of fuzzy rules so that the task of projecting new points becomes a trivial job. We assume that the data set under consideration is obtained from a time invariant probability distribution. Under this assumption if we extract the rule base from a *representative* sample X then its performance on a new data point $\mathbf{x}_k$ is expected to be nearly the same as that of the system identified from the data set $X' = X \cup \{\mathbf{x}_k\}$. So our scheme consists of the following three major steps:

1) project X by Sammon's algorithm to generate Y;
2) extract a fuzzy rule base R from (X, Y) as described next;
3) use R to project any new data point.

Let the input data set be $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathcal{R}^p$ and projected (output) data set be $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\} \subset \mathcal{R}^q$. We define

$$X^* = \left\{ \mathbf{x}_i^* = \begin{pmatrix} \mathbf{x}_i \in \mathcal{R}^p \\ \mathbf{y}_i \in \mathcal{R}^q \end{pmatrix} \in \mathcal{R}^{p+q}, \qquad i = 1, 2, \ldots, n \right\}$$

[1]Loosely, we call a data set linear if a linear transformation can project it to a lower dimension preserving the cluster structure in the original data. Otherwise, the data set is nonlinear in nature.

TABLE III
Epochs Used for Various Methods for Every Data Set

SAM	100
PCA	10000
ONN	10000
MNN	10000(PCA)+1000(MLP)+10000(Sammon net)
FRTS	100(SAM)+100(FCM)+10000(TS)
FRMA	100(SAM)+100(FCM)+10000(MA)

TABLE IV
Sammon Errors on X_{Tr}, and on X When Training was Done on X_{Tr}

Data Set	Methods	$SE(R_{X_{Tr}}, X_{Tr})$	$SE(R_{X_{Tr}}, X)$
IRIS	SAM	**0.004653**	**0.004693**
	PCA	0.015391	0.012137
	ONN	0.028049	0.029450
	MNN	0.007553	0.008097
	FRTS	0.010243	0.226525
	FRMA	0.010380	0.102227
Helix	SAM	**0.001192**	**0.001415**
	PCA	0.002205	0.002041
	ONN	0.019336	0.020057
	MNN	0.003249	0.003082
	FRTS	0.002816	0.001502
	FRMA	0.001985	0.002140
Elongated Clusters	SAM	**0.000752**	**0.000924**
	PCA	0.019768	0.018788
	ONN	0.049553	0.051860
	MNN	0.006700	0.006257
	FRTS	0.000776	0.000811
	FRMA	0.001257	0.001219
Sphere Shell	SAM	**0.021783**	**0.038214**
	PCA	0.104631	0.103091
	ONN	0.387898	0.548430
	MNN	0.040100	0.046007
	FRTS	0.023808	0.032445
	FRMA	0.024524	0.031867
10-D Normal Mixture	SAM	**0.013757**	**0.015138**
	PCA	0.038642	0.038246
	ONN	0.113191	0.117386
	MNN	0.117438	0.121100
	FRTS	0.019950	0.023553
	FRMA	0.065397	0.062955

i.e., $\mathbf{x}_i^*$ is $\mathbf{x}_i$ augmented by $\mathbf{y}_i$. Suppose we cluster X^* by some clustering algorithm producing a set of centroids

$$V^* = \left\{ \mathbf{v}_i^* = \begin{pmatrix} \mathbf{v}_i^x \in \mathcal{R}^p \\ \mathbf{v}_i^y \in \mathcal{R}^q \end{pmatrix} \in \mathcal{R}^{p+q}, \qquad i = 1, 2, \ldots, c \right\}$$

and a partition matrix (hard or fuzzy). This clustering results can be used to extract fuzzy rules. Use of clustering 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 if 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 $\mathbf{v}_i^y$. Thus, if $\mathbf{v}_i^*$ 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 a rough indication that such a cluster corresponds to a locally continuous and smooth input–output relation. Even when the data do not have any cluster structure in the pattern recognition sense, it is possible to partition the input–output data into several subsets such that for each subset such a rule can be written. Thus, the ith cluster can be translated into a rule of the form

Mamdani–Assilian (MA) model [14] :
If $\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$ then $\mathbf{y}$ is CLOSE to $\mathbf{v}_i^y$
Takagi–Sugeno (TS) model [15] :
If $\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$ then $\mathbf{y} = \mathbf{u}_i(\mathbf{x}, \mathbf{v}_i^y)$.

Usually, the antecedent part, if $\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$, is written as a conjunction of p atomic clauses: If x_1 is CLOSE to v_{i1}^x and x_2 is CLOSE to v_{i2}^x and $\cdots$ and x_p is CLOSE to v_{ip}^x. The function $\mathbf{u}_i(.)$ in the Takagi–Sugeno (T–S) case primarily models the behavior of the input–output relation in the neighborhood of $\mathbf{v}_i^y$. Rules can also be generated by clustering Y. In this case, the centers $\{\mathbf{v}_i^x\}$ can be obtained as centroids of associated clusters in X. Similarly, when X is clustered, the centers $\{\mathbf{v}_i^y\}$ can be generated as centroids of associated clusters in Y.

If a fuzzy clustering algorithm is used, then we can induce fuzzy clusters on different axes by projecting the membership values of the extracted clusters. Suppose the clustering is done in X^*. One of the simple ways to assign a membership value to the input vector $\mathbf{x}_i$ is by

$$\mu(\mathbf{x}_i) = \max_j \left\{ \mu(\mathbf{x}_j^*) = \mu \begin{pmatrix} \mathbf{x}_j = \mathbf{x}_i \\ y_j \end{pmatrix}, \qquad \mathbf{x}_j^* \in X^* \right\}.$$

And then each component of $\mathbf{x}_i \in \mathcal{R}^p$ can be assigned membership values in the same way, i.e.,

$$\mu(x_{ij}) = \max_k \{\mu(\mathbf{x}_k) | x_{kj} = x_{ij}\}, \qquad \forall j = 1, 2, \ldots, p.$$

Although several authors [16]–[19] have used exploratory data analysis for rule based system identification, there are several problems as discussed in Pal *et al.* [25] which need careful attention to make such approaches useful. We next discuss these issues [25] and our solutions to them.

1) **Choice of the clustering algorithm**: Although there could be many choices, we use the FCM algorithm as we have no idea about the type of cluster structure that may be present in the data. FCM extracts hyperspherical clusters and any input–output relation, even say a linear relation, can be approximated by a reasonable number of hyperspherical patches (clusters).
2) **Choice of the clustering domain**: There are four choices: clustering of X, clustering of Y, clustering of X^*, or clustering of both X and Y separately. In this paper, we decided to use X^* because in X^*, $\mathbf{x}$ and the corresponding $\mathbf{y}$ are tied together so that interpretation of a cluster as a rule becomes easier.
3) **Deciding the number of rules or clusters**: Researchers have used different cluster validity indexes like the Xie–Beni [20] index, Gath–Geva [21] index and so on. These indexes have been developed for cluster validation without paying attention to the rule-based system identification problem. Hence, use of these indexes for the present problem is *debatable*. Suppose we have two compact big (the convex hull of each cluster is big) well separate clusters. Here any cluster validity index will indicate $c = 2$, but each cluster has too high variability to be modeled by a single rule. We have heuristically decided the number of clusters.
4) **Choice of the structure of the rule base**: This refers to deciding on whether Mamdani–Assilian (MA) model or TS model is to be used. We do not have any prior knowledge about the possible nonlinear relation that might be present. So we decided to use both the MA model and the TS model with linear consequents. Under a fairly general set of conditions, the MA model can approximate any nonlinear relation, similarly, TS model with linear consequents can also approximate any nonlinear relation.
5) **Estimation of parameters of the model**: We shall discuss it in the appropriate place.
6) **Validation of the model**: A common practice is to use square error on the training data as an index for validation. In the present case, in addition to visual assessment of outputs we also consider Sammon's error for validation of the system. Moreover, we partition the data X into X_{Tr} and X_{Te} such that $X = X_{Tr} \cup X_{Te}$ and $X_{Tr} \cap X_{Te} = \phi$. We identify the rule base $R_{X_{Tr}}$ using X_{Tr} and test $R_{X_{Tr}}$ on X_{Te}. But how do we assess the quality of output with X_{Te}? Let $SE(R, X)$ be the Sammon error on the data set X obtained by the rule base R. To validate the model, we compare $SE(R_{X_{Tr}}, X)$ with Sammon error directly computed on X by the original Sammon's algorithm.

Each of these issues requires a much detailed and careful analysis and that is beyond the scope of this paper. We mentioned these issues here to indicate that the proposed system can further be improved by paying closer attention to these issues. Here, we simply establish the utility of fuzzy rule-based system for dimensionality reduction. For the sake of completeness, we next briefly discuss the FCM algorithm.

Given $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathcal{R}^p$, the FCM algorithm finds a partition matrix $U = [u_{ik}]_{c \times n}$ and a set of centroids $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$ minimizing [24]

$$J_m(U, V) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|_A^2 \tag{4}$$

where m is a weighting exponent greater than 1 (typically $m = 2$) and the inner product norm metric

$$\|\mathbf{x}_k - \mathbf{v}_i\|_A^2 = (\mathbf{x}_k - \mathbf{v}_i)^T A (\mathbf{x}_k - \mathbf{v}_i)$$

c is the number of clusters, and A is any $p \times p$ positive–definite matrix.

Let $\mathbf{v}_i^z$, $i = 1, 2, \ldots, c$ be the centroids of the clusters obtained by FCM on X^z. We translate the ith cluster (using the TS model [15]) into a rule of the form: R_i^{TS}: If $\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$ then $\mathbf{y} = \mathbf{u}_i(\mathbf{x}, \mathbf{v}_i^y)$. Note that, "$\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$" is an antecedent clause with p components. Thus, R_i^{TS}: If x_1 is CLOSE to v_{i1}^x and $\cdots$ and x_p is CLOSE to v_{ip}^x then $\mathbf{y} = \mathbf{u}_i(\mathbf{x}, \mathbf{v}_i^y)$. Since $\mathbf{y} \in \mathcal{R}^q$, R_i can also be viewed as q different rules, one for each component of $\mathbf{y}$. This set of c rules forms an initial rule base for data projection. For the MA model we translate the ith cluster as R_i^{MA}: if $\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$ then $\mathbf{y}$ is CLOSE to $\mathbf{v}_i^y$. Like a TS rule, here too the antecedent part is written as a conjunction of p atomic clauses.

For an input vector $\mathbf{x}_k \in \mathcal{R}^p$, let α_i be the firing strength of the rule R_i^{TS} computed using any T-norm [26], say product. Then $\hat{\mathbf{y}}_k = (\hat{y}_{k1}, \hat{y}_{k2}, \ldots, \hat{y}_{kq})^T$ is computed as

$$\hat{\mathbf{y}}_k = \frac{\sum_{i=1}^{c} \alpha_i.\mathbf{u}_i(\mathbf{v}_i^y)}{\sum_{i=1}^{c} \alpha_i}. \tag{5}$$

For the MA model we can compute $\hat{\mathbf{y}}_k$ using one of several defuzzification methods like the height method, the center of gravity method [26]. For computational simplicity we use the height method of defuzzification [26].

A. The Rule Identification Scheme

As mentioned earlier R_i^{TS} corresponds to a set of q rules. We denote these q rules as R_{ij}^{TS}: If $\mathbf{x}$ is CLOSE to $\mathbf{v}_i^x$ then $y_j = u_{ij}(\mathbf{x}, v_{ij}^y)$, $j = 1, 2, \ldots, q$. We use $u_{ij} = d_{ij0}.v_{ij}^y + d_{ij1}.x_1 + \cdots + d_{ijp}.x_p$, $i = 1, 2, \ldots, c$; $j = 1, 2, \ldots, q$, where $d_{ijl} (l = 0, 1, \ldots, p)$ are constants to be identified. Thus, the output is a linear combination of v_{ij}^y and the input. Since v_{ij}^y is given (i.e., a constant), without loss of generality we denote $d_{ij0}.v_{ij}^y$ as d_{ij0} and estimate it along with other parameters. Hence, the output is computed by

$$y_{kj} = \frac{\sum_{i=1}^{c} \alpha_i.u_{ij}(\mathbf{x}_k, v_{ij}^y)}{\sum_{i=1}^{c} \alpha_i}, \qquad j = 1, 2, \ldots, q. \tag{6}$$

As pointed out by Takagi and Sugeno, given a set of rules with fixed antecedents, optimizing the parameters in the consequent equations with respect to training data reduces to a linear least square error estimation. This problem can be solved easily and the solution is always globally optimal [15]. Since the output cluster centroids are not necessary for this model, one might be inclined to use only clustering of the input data. However, it is not a good idea as a cluster found in X ignoring Y may correspond to more than one region of the output space.

To choose the appropriate set of consequent parameters d_{ijl}, we have to formulate the optimization problem as a linear least squares problem. In doing this, we rewrite (6) as

$$y_{kj} = \sum_{i=1}^{c} \mu_{ki} \cdot (d_{ij0} + d_{ij1} \cdot x_{k1} + \cdots + d_{ijp} \cdot x_{kp})$$

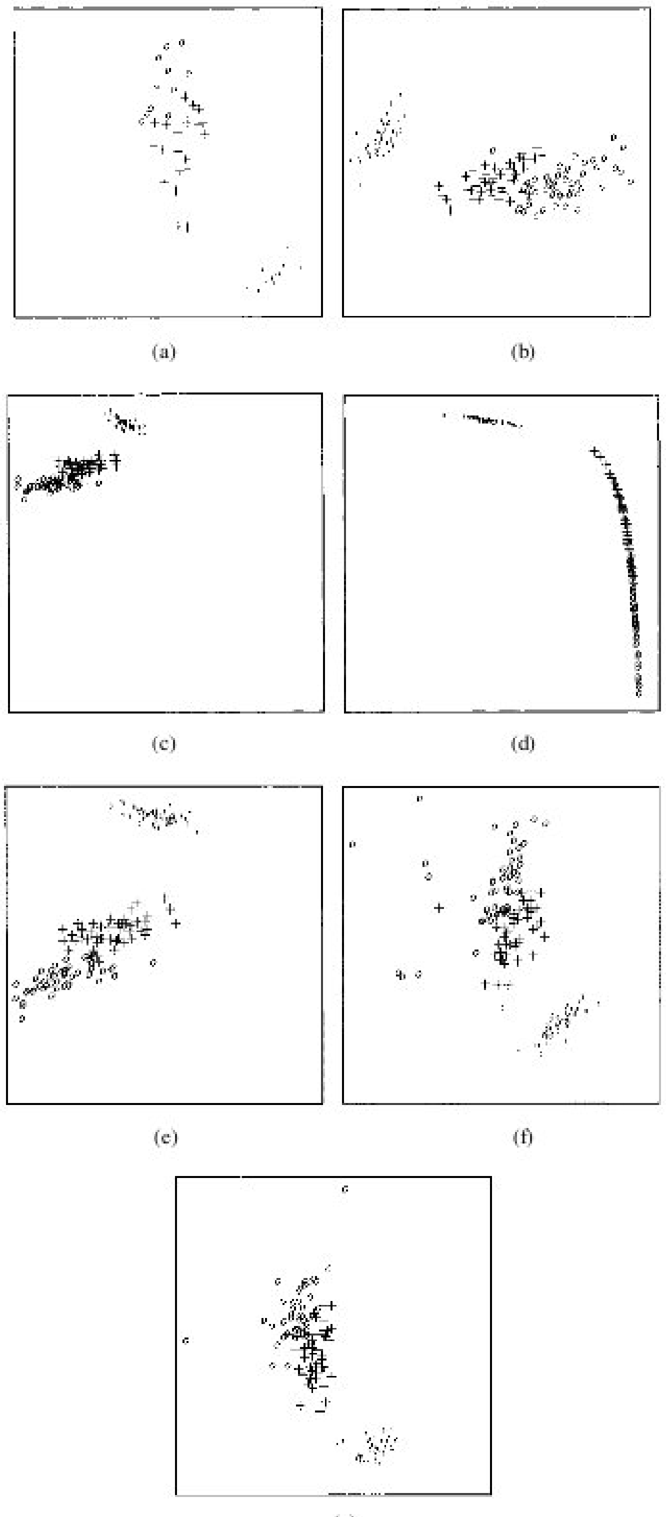

Fig. 2. Results for Iris. (a) SAM output for Iris on X_{Tr}. (b) SAM output for Iris on X. (c) PCA output for Iris with training on X_{Tr} and test on X. (d) ONN output for Iris with training on X_{Tr} and test on X. (e) MNN output for Iris with training on X_{Tr} and test on X. (f) FRTS output for Iris with training on X_{Tr} and test on X. (g) FRMA output for Iris with training on X_{Tr} and test on X.

where

$$\mu_{ki} = \frac{\alpha_i}{\sum_{i=1}^{c} \alpha_i}.$$

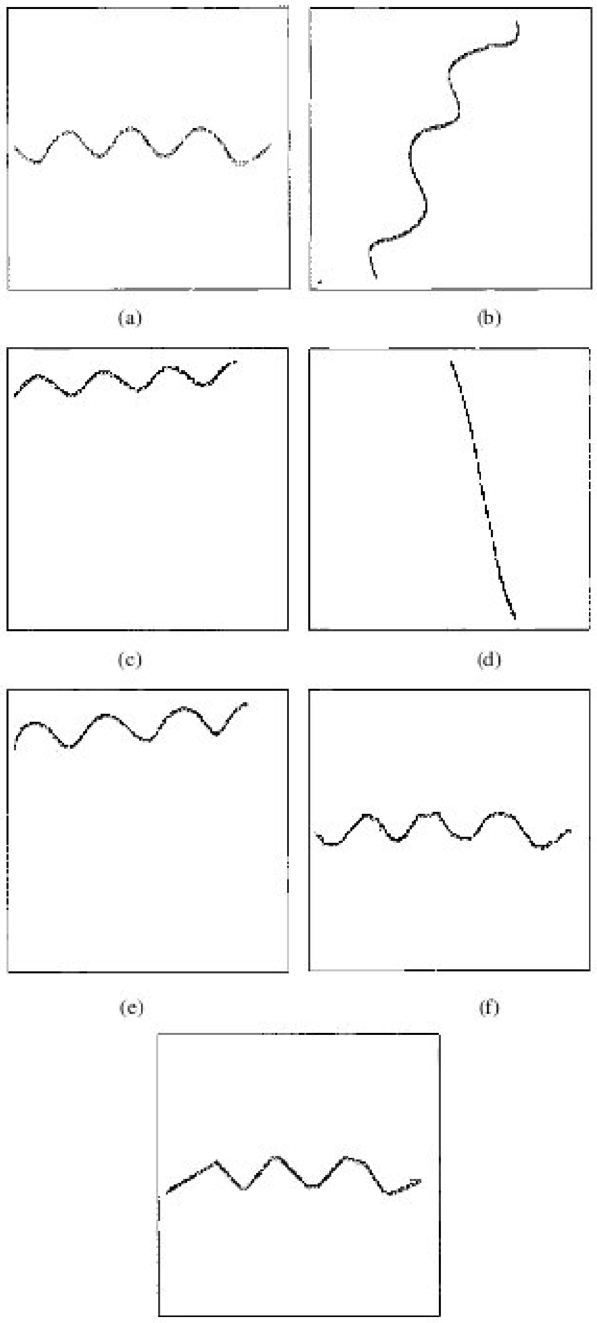
(a) (b) (c) (d) (e) (f) (g)

Fig. 3. Results for helix. (a) SAM output for helix on X_{Tr}. (b) SAM output for helix on X. (c) PCA output for helix with training on X_{Tr} and test on X. (d) ONN output for helix with training on X_{Tr} and test on X. (e) MNN output for helix with training on X_{Tr} and test on X. (f) FRTS output for helix with training on X_{Tr} and test on X. (g) FRMA output for helix with training on X_{Tr} and test on X.

Therefore

$$y_{kj} = \sum_{i=1}^{c} (d_{ij0} \cdot \mu_{ki} + d_{ij1} \cdot x_{k1} \cdot \mu_{ki} + \cdots + d_{ijp} \cdot x_{kp} \cdot \mu_{ki}). \quad (7)$$

Then using a set of input–output data, we can easily obtain the consequent parameters

$$D^j = (d_{1j0} d_{2j0} \cdots d_{cj0} d_{1j1} d_{2j1} \cdots d_{cj1} \cdots d_{1jp} d_{2jp} \cdots d_{cjp})^T, \quad j = 1, 2, \ldots, q$$

by the least square error method using (7). So, for a set of n input–output data, (7) gives us a system $Y = A \cdot D$ where $Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)^T$, $D = (D^1 D^2 \ldots D^q)_{(p-1)c \times q}$ and

$$A = \begin{pmatrix} \mu_{11} \cdots \mu_{1c} & \mu_{11} \cdot x_{11} \cdots \mu_{1c} \cdot x_{11} & \cdots & \mu_{11} \cdot x_{1p} \cdots \mu_{1c} \cdot x_{1p} \\ \mu_{21} \cdots \mu_{2c} & \mu_{21} \cdot x_{21} \cdots \mu_{2c} \cdot x_{21} & \cdots & \mu_{21} \cdot x_{2p} \cdots \mu_{2c} \cdot x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{n1} \cdots \mu_{nc} & \mu_{n1} \cdot x_{n1} \cdots \mu_{nc} \cdot x_{n1} & \cdots & \mu_{n1} \cdot x_{np} \cdots \mu_{nc} \cdot x_{np} \end{pmatrix}.$$

Here, $Y_{n \times q}$ is a matrix of output values, $A_{n \times c(p+1)}$ is a constant matrix as all x_{ij} and μ_{ij} are given, $D_{c(p+1) \times q}$ is the matrix which contains all parameters to be estimated. The estimate of D minimizing this square error $\| AD - Y \|^2$ is given by

$$D = (A^T A)^{-1} A^T Y. \quad (8)$$

B. Choice of Antecedent Memberships

In order to implement the rule base we need to define the membership function for "x_j CLOSE to v_{ij}^x." We use *asymmetric* triangular functions having peak, $a_{ij} = v_{ij}^x$ and widths b_{ij}^L, b_{ij}^R (here L and R indicate the left and right widths of the triangle). Note that, a_{ij}s, b_{ij}^Ls and b_{ij}^Rs for all q rules (one for each output variable) corresponding to a particular cluster i are the same. For the jth feature, to find b_{ij}^L and b_{ij}^R we proceed as follows. We sort v_{ij}^x, $i = 1, 2, \ldots, c$. Let the sorted list be $v_{i_l j}^x$, $l = 1, 2, \ldots, c$. Suppose v_{ij}^x takes the mth position in the sorted list, i.e., $v_{i_m j}^x = v_{ij}^x$, then the width of the fuzzy set associated to v_{ij}^x, (i.e., the mth fuzzy set on the axis for jth feature) is defined by

$$b_{m,j}^L = \left\{ v_{i_m j}^x - v_{i_{m-1} j}^x \right\}, \qquad m = 2, \ldots, c-1 \quad (9)$$

and

$$b_{m,j}^R = \left\{ v_{i_{m+1} j}^x - v_{i_m j}^x \right\}, \qquad m = 2, \ldots, c-1. \quad (10)$$

The widths of the fuzzy sets at the two extreme ends of the domain of x_j are defined by (11)–(14) as follows:

$$b_{1,j}^L = \left\{ v_{i_1 j}^x - (L_j - (0.05 * (H_j - L_j))) \right\} \quad (11)$$

$$b_{1,j}^R = \left\{ v_{i_2 j}^x - v_{i_1 j}^x \right\} \quad (12)$$

$$b_{c,j}^L = \left\{ v_{i_c j}^x - v_{i_{c-1} j}^x \right\} \quad (13)$$

$$b_{c,j}^R = \left\{ (H_j + (0.05 * (H_j - L_j))) - v_{i_c j}^x \right\}. \quad (14)$$

Here, L_j and H_j are the lowest and highest values of feature j. Equations (11) and (14) expands the domain of the jth feature by 5% on either side. The choice of the asymmetric triangular membership functions with the left and right widths as defined in (9)–(14) ensures sufficient overlaps between adjacent membership functions. Table I shows the ten six-dimensional (four-input and two-output variables) centroids obtained from the FCM algorithm for the normalized IRIS data [22]; while Table II displays the sorted values corresponding to column x_1 of Table I. For this feature, $H_1 = 0.910\,256$ and $L_1 = 0.589\,744$. This makes $b_{1,1}^L = 0.037\,881\,6$ and $b_{10,1}^R = 0.038\,881\,62$. Columns 3 and 4 of Table II display the left end and right end of all ten membership functions defined

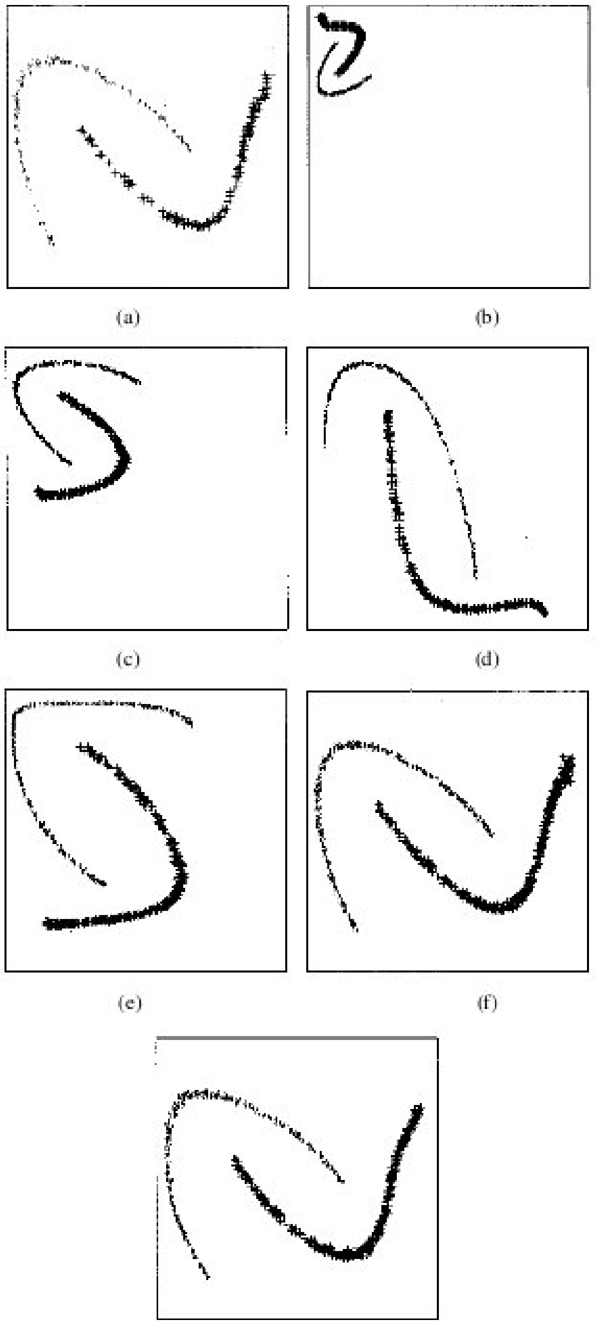

Fig. 4. Results for elongated clusters. (a) SAM output for elongated clusters on X_{Tr}. (b) SAM output for elongated clusters on X. (c) PCA output for elongated clusters with training on X_{Tr} and test on X. (d) ONN output for elongated clusters with training on X_{Tr} and test on X. (e) MNN output for elongated clusters with training on X_{Tr} and test on X. (f) FRTS output for elongated clusters with training on X_{Tr} and test on X. (g) FRMA output for elongated clusters with training on X_{Tr} and test on X.

on x_1 and Fig. 1 depicts the graphs of the corresponding ten membership functions.

C. Further Tuning of Parameters

In Section IV-A, we have obtained the least square error (LSE) estimate of the consequent parameters of a TS model assuming fixed values for the antecedent parameters. We take this as an *initial* choice for the consequent and then use the gradient descent method to further refine *all* parameters (including antecedent membership functions) because the objective function minimized by the clustering algorithm and the objective function that a rule base should minimize are not the same. Moreover, such a two-stage hybrid scheme for further tuning of consequents along with membership parameters is justified because when membership parameters are altered the LSE estimate of the consequents may (usually will) not remain optimal. The rule base parameters a_{ij}, b_{ij} and d_{ijl} are tuned using gradient descent to minimize the error $E = \sum_{k=1}^{n} \|\hat{\mathbf{y}}_k - \mathbf{y}_k\|^2$ as

$$a_{ij}(t+1) = a_{ij}(t) - \alpha_a \frac{\partial E}{\partial a_{ij}} \tag{15}$$

$$b_{ij}^{L}(t+1) = b_{ij}^{L}(t) - \alpha_b \frac{\partial E}{\partial b_{ij}^{L}} \tag{16}$$

$$b_{ij}^{R}(t+1) = b_{ij}^{R}(t) - \alpha_b \frac{\partial E}{\partial b_{ij}^{R}} \tag{17}$$

and

$$d_{ijl}(t+1) = d_{ijl}(t) - \alpha_d \frac{\partial E}{\partial d_{ijl}}. \tag{18}$$

In (15)–(18), t indicates the iteration number. One might argue that direct gradient descent (without the LSE estimate) on all parameters should be able to produce the same result. Yes, theoretically it is possible. But we all know the problems with gradient descent when we do not have a good starting solution. In this hybrid scheme, since the initial antecedent memberships are judiciously chosen based on cluster analysis, they are likely to form a reasonable good set of linguistic values. The LSE estimate of the consequent parameters keeping the antecedent memberships fixed will result in a fairly good rule based system. Hence, our gradient descent step will start with a good initialization and we are likely to get a better rule base with less training epochs. For the MA model, we tune all parameters of the antecedent membership functions and the peak of the consequent membership functions using gradient descent.

To summarize we get two schemes: fuzzy rules extracted by TS model (FRTS) and fuzzy rules extracted by MA model (FRMA).

```
FRTS( )
{
  Run Sammon's projection with $X \subset \mathcal{R}^p$ as
  input to generate $Y \subset \mathcal{R}^q$
  Augment $X$ by $Y$ to get $X^*$;
  Run FCM algorithm to produce $c$ clusters
  using $X^*$;
  Define MF's for antecedents using
  (9)-(14);
  Form the initial rule base;
  Find LSE estimate of the consequent pa-
  rameters using (8);
  /* Tuning of the rule base */
  Refine the rule base using equations
  (15)-(18) till
    RMS error $< \epsilon$ or iteration = maxsteps;
}
```

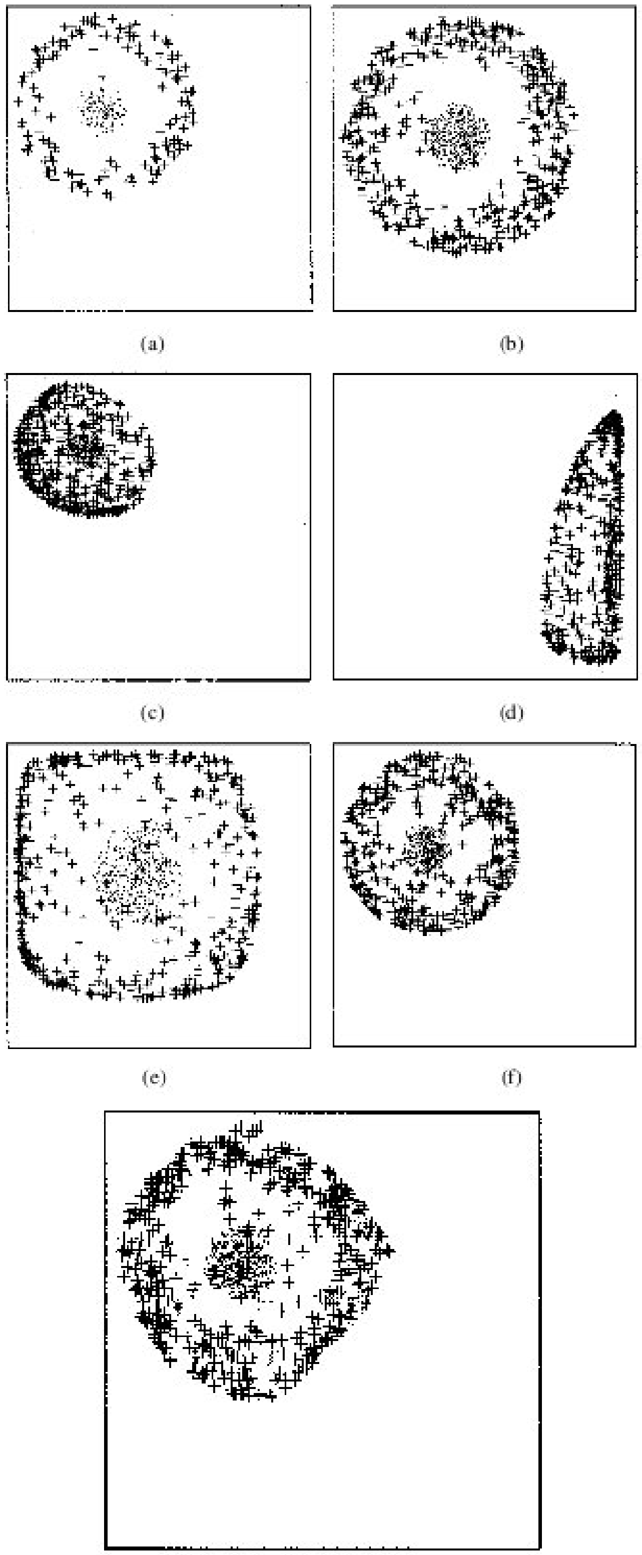

Fig. 5. Results for sphere shell. (a) SAM output for sphere shell on X_{Tr}. (b) SAM output for sphere shell on X. (c) PCA output for sphere shell with training on X_{Tr} and test on X. (d) ONN output for sphere shell with training on X_{Tr} and test on X. (e) MNN output for sphere shell with training on X_{Tr} and test on X. (f) FRTS output for sphere shell with training on X_{Tr} and test on X. (g) FRMA output for sphere shell with training on X_{Tr} and test on X.

In Algorithm FRTS, ϵ is a preassigned small positive quantity and maxsteps is a predetermined limit on the maximum number of iterations. We can now use this trained rule base to project X and any new data points.

```
FRMA( )
{
  Run Sammon's projection with $X \subset \mathcal{R}^p$ as
  input to generate $Y \subset \mathcal{R}^q$
  Augment $X$ by $Y$ to get $X^*$;
  Run FCM algorithm to produce $c$ clusters
  using $X^*$;
  Define MF's for antecedents using
  (9)-(14);
  Form the initial rule base;
  /* Tuning of the rule base */
  Refine the rule base using equations
  (15)-(18) till
    RMS error $< \epsilon$ or iteration $=$ maxsteps;
}
```

This trained rule base can now be used to project X and any new data points.

V. Implementation and Results

A. Data Sets and Computational Protocols

To demonstrate the effectiveness of the proposed scheme we implemented the following algorithms: Sammon's algorithm (SAM), principal component analysis (PCA) net, original NN implementation of Jain and Mao (ONN), modified NN implementation of Jain and Mao (MNN) that uses an MLP trained to learn the principal components as the initialization of the net for data projection, FRTS and FRMA. All algorithms are tested on five data sets named **Iris**, **Helix**, **Sphere-Shell**, **Elongated-Clusters**, and **10-D Normal-mixture**.

Iris [22] is a well-known data set consisting 150 points from three classes in a four-dimensional space. Each class has 50 points. One of the classes is well separated from the rest while the other two have some overlap.

Helix is a three-dimensional (3-D) data containing 1000 points drawn uniformly distributed on a helix [3].

Sphere-shell [23] is a synthetic data set consisting of 1000 points in 3-D. 500 points are selected randomly within a hemisphere of radius $r1$ and rest 500 are generated in a shell defined by two hemispheres of radii $r2$ and $r3$, such that $r1 < r2 < r3$.

Elongated-cluster [9] is also a synthetic data set consisting of two elongated clusters of 500 points each in 3 space.

Ten-dimensional (10-D) normal-mixture is synthetic data containing a mixture of three ten-variate normals. It contains 200 points from each of the normal distributions.

For ONN and MNN we used networks with one hidden layer having 20 nodes. For FRTS and FRMA rule-based systems, we used ten rules for all data sets. In all cases except SAM, we used randomly selected 30% of the data set for training and the entire data set for testing of the systems. We applied SAM on the entire data set so that we can compare the generalization of the identified system. The number of epochs for each method is listed in Table III.

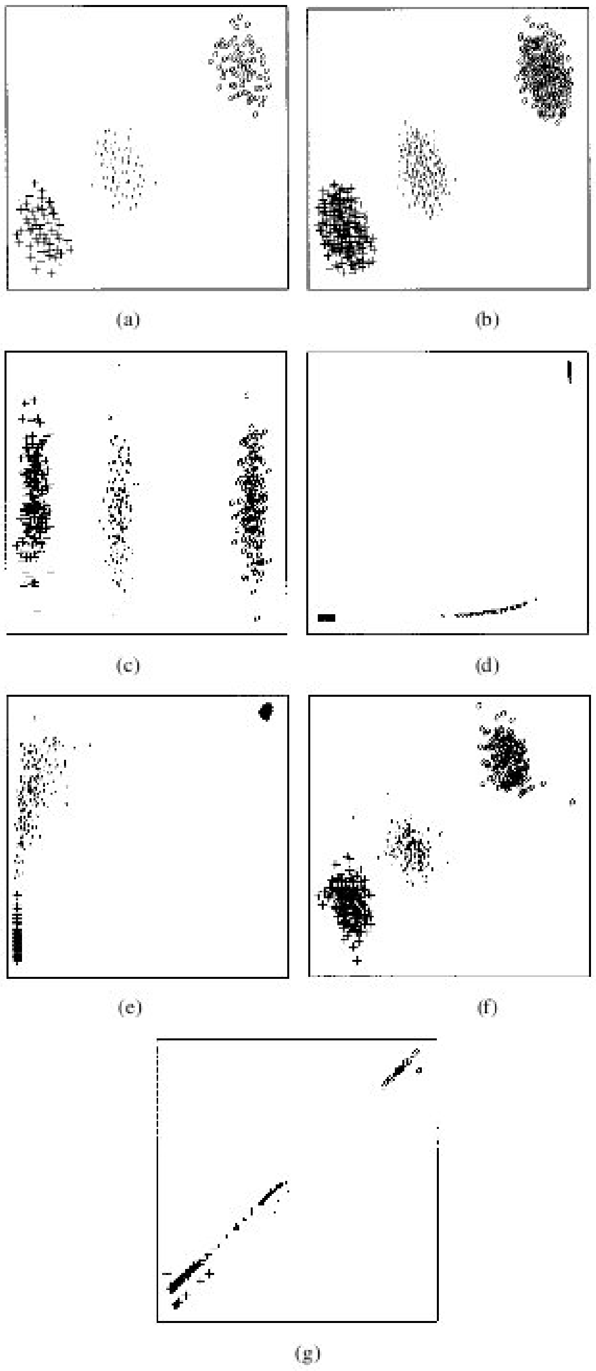

Fig. 6. Results for 10-D normal mixture. (a) SAM output for 10-D normal mixture on X_{Tr}. (b) SAM output for 10-D normal mixture on X. (c) PCA output for 10-D normal mixture with training on X_{Tr} and test on X. (d) ONN output for 10-D normal mixture with training on X_{Tr} and test on X. (e) MNN output for 10-D normal mixture with training on X_{Tr} and test on X. (f) FRTS output for 10-D normal mixture with training on X_{Tr} and test on X. (g) FRMA output for 10-D normal mixture with training on X_{Tr} and test on X.

B. Results

Table IV reports $SE(R_{X_{Tr}}, X_{Tr})$ and $SE(R_{X_{Tr}}, X)$ for all data sets. In addition, the first row for each data set shows (in **bold**) the Sammon's error obtained by SAM on X_{Tr} and X. Column 3 corresponds to Sammon's error on X_{Tr} when the system is trained on X_{Tr}; while column 4 corresponds to Sammon's error on the entire data set X when the training is done on X_{Tr}.

For visual assessment of the results, we display the scatterplots of the projected data. Fig. 2 includes seven scatterplots of the two dimensional outputs produced for IRIS data. In Fig. 2 and in all other figures the following numbering schemes are used: (a) corresponds to the scatterplot of the output produced by SAM on X_{Tr}; (b) corresponds to the scatterplot of the output produced by SAM on X; (c)–(g), respectively, correspond to the output produced by PCA, ONN, MNN, FRTS and FRMA for the entire data set X when the training is done only on X_{Tr}, i.e., they correspond to column 4 of Table IV.

For IRIS except ONN, all methods produced good projections. For ONN, although the scatterplot looks quite different from the rest, it does separate the three classes. $SE(R_{X_{Tr}}, X)$ is the highest for FRTS, though visually the output is quite good. This tells that the rule base $R_{X_{Tr}}$ is not doing a good job of optimizing Sammon's error on X but, since it did a good job for X_{Tr}, i.e., $R_{X_{Tr}}$ has captured the structure inherent in the data, the projection of X is reasonably good.

Fig. 3 displays scatterplots of the projected data for Helix. Note that, the orientation of the projections by SAM are different for X_{Tr} and X [Fig. 3(a) and (b)], but the structural content remains the same. For Helix the performance of ONN is quite poor. Table IV shows that in terms of SE (on X_{Tr} and X), again ONN is the worst. The scatterplots also reveal the same.

For elongated clusters (Fig. 4) and Sphere-shell (Fig. 5) the TS rule base (FRTS) does the best job while again ONN is the worst. For both data sets $SE(R_{X_{Tr}}, X_{Tr})$ with FRTS is almost the same as that of Sammon's error. Similarly, $SE(R_{X_{Tr}}, X)$ with FRTS is very close to the Sammon error directly computed on X by Sammon's original method. This reveals that FRTS does an excellent job of generalization (i.e., achieves a good prediction capability). For elongated clusters although the SE value for FRMA is not as good as that of FRTS, but both of them are much smaller than the SE for other methods tried. Fig. 5 shows that for sphere-shell FRTS results in the best output which not only preserves the shapes of the two classes, but it also nicely separates the two.

Fig. 6 depicts the scatterplots for the 10-D normal mixture data. In this case, PCA does a very good job (both in terms of SE and scatterplots). None of ONN and MNN could do a good job of projection at least for the simulations that we tried. On the other hand, both FRTS and FRMA exhibit good generalization and structure preservation.

VI. Conclusion

In this paper, we have proposed a fuzzy rule-based scheme for structure preserving dimensionality reduction (feature extraction). It is based on the structure preserving characteristic of Sammon's method and the generalization capability of rule-based fuzzy systems. We used both the TS model with consequent expressed as a linear combination of the input variables, as well as, the MA model with height method of defuzzification. An initial rule base was extracted using cluster analysis. For the TS model, the consequent parameters of these rules are then estimated using LSE technique. The antecedent as well as the consequent parameters of the rule base thus obtained are further refined using gradient descent. For the MA model we tuned both antecedent and consequent parameters using gradient search. We tested the proposed schemes on several data sets and obtained excellent results. Our method achieved four things: 1) unlike Sammon's method it has good predictability; 2) computationally, it is more efficient than original Sammon's method; 3) it has better predictability than some of the NN implementations of Sammon's method; and 4) it can detect outliers while testing.

References

[1] K. Fukunga, *Introduction to Statistical Pattern Recognition*, 2nd ed. New York: Academic, 1990.

[2] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ: Prentice-Hall, 1988.

[3] J. W. Sammon, Jr., "A nonlinear mapping for data structure analysis," *IEEE Trans. Comput.*, vol. C-18, pp. 401–409, 1969.

[4] B. Schachter, "A Nonlinear mapping algorithm for large databases," *Comput. Graph. Image Process.*, vol. 7, pp. 271–278, 1978.

[5] C. E. Pykett, "Improving the efficiency of Sammon's nonlinear mapping by using clustering archetypes," *Electron. Lett.*, vol. 14, pp. 799–800, 1980.

[6] C. L. Chang and R. C. T. Lee, "A heuristic relaxation method for nonlinear mapping in cluster analysis," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-3, pp. 197–200, 1973.

[7] G. Biswas, A. K. Jain, and R. C. Dubes, "Evaluation of projection algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 701–708, 1981.

[8] A. K. Jain and J. Mao, "Artificial neural networks for nonlinear projection of multivariate data," in *Proc. IEEE Int. Joint Conf. Neural Networks*, vol. 3, 1992, pp. 59–69.

[9] 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, Apr. 1995.

[10] N. R. Pal and V. K. Eluri , "Neural networks for dimensionality reduction," in *Progress in Connectionist-Based Information Systems, Proc. 4th Int. Conf. Neural Inform. Processing*, vol. 1, Kasabov, Ed., New Zealand, 1997, pp. 221–224.

[11] ——, "Two efficient connectionist schemes for structure preserving dimensionality reduction," *IEEE Trans. Neural Networks*, vol. 9, pp. 1142–1154, Dec. 1998.

[12] J. Rubner and P. Tavan, "A self-organizing network for principal component analysis," *Europhys. Lett.*, vol. 10, pp. 693–698, 1989.

[13] J. Rubner and K. Schulten, "Development of feature detectors by self organization," *Biol. Cybern.*, vol. 62, pp. 193–199, 1990.

[14] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Mach. Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[15] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Feb. 1985.

[16] M. Delgado, A. F. Gomez-Skarmeta, and F. Martin, "A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 223–233, Apr. 1997.

[17] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.

[18] 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.

[19] R. Rovatti and R. Guerrieri, "Fuzzy sets of rules for system identification," *IEEE Trans. Fuzzy Syst.*, vol. 4, pp. 89–102, Apr. 1996.

[20] X. L. Xie and G. A. Beni, "Validity measure for fuzzy clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 841–846, Aug. 1991.

[21] T. Gath and A. B. Gava, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 11, pp. 773–781, July 1989.

[22] E. Anderson, "The IRISes of the Gaspe peninsula," *Bull. Amer. IRIS Soc.*, vol. 59, pp. 2–5, 1935.

[23] H. Niemann, "Linear and nonlinear mapping patterns," *Pattern Recogn.*, vol. 12, pp. 83–87.

[24] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[25] N. R. Pal, K. Pal, J. C. Bezdek, and T. Runkler, "Some issues in system identification using clustering," in *Proc. Int. Conf. Neural Networks, ICNN*. Piscataway, NJ, 1997, pp. 2524–2529.

[26] D. Driankov, H. Hellendorn, and M. Reinfrank, *An Introduction to Fuzzy Control*. New York: Springer-Verlag, 1993.

Nikhil R. Pal (M'91–SM'00) received the B.Sc. degree in physics (with honors) and the M.B.M. degree, both from the University of Calcutta, India, in 1979 and 1982, respectively, and the M.Tech. and Ph.D. degrees (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, Calcutta. From September 1991 to February 1993, July 1994 to December 1994, October 1996 to December 1996, and January to July 2000, he visited the Computer Science Department of the University of West Florida, Pensacola. He was also a Guest Faculty Member of the University of Calcutta. He coauthored *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Boston, MA: Kluwer, 1999), coedited a volume *Advances in Pattern Recognition and Digital Techniques, ICAPRDT99* (Narosa, New Delhi), and *Pattern Recognition in Soft Computing Paradigm* (Singapore: World Scientific, 2001). He is an Associate Editor of the *International Journal of Fuzzy Systems* and the *International Journal of Approximate Reasoning*, and is also an Area Editor of *Fuzzy Sets and Systems*. His research interests include image processing, pattern recognition, fuzzy sets theory, measures of uncertainty, neural networks, genetic algorithms, and fuzzy logic controllers.

Dr. Pal is an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS—B.

Vijaya Kumar Eluri received the B.Tech. degree in computer science from Andhra University, India, and the M.Tech. degree in computer science from the Indian Statistical Institute, Calcutta, in 1993 and 1996, respectively.

Currently, he is working as a Senior Software Engineer at HNC Software, Inc., San Diego, CA. His research interests includes neural networks, fuzzy logic, dimensionality reduction, and pattern recognition.

Gautam K. Mandal received the M.Sc. degree in applied mathematics and the M.Tech. degree in atmospheric sciences, both from the University of Calcutta, India.

Previously, he was with the Center for Atmospheric Sciences, University of Calcutta and Surendranath College, Calcutta, India. He is currently with Cap Gemini Ernst and Young Singapore Pte., Ltd., Singapore.