The Third AFSS(1998), 426−431

# Structure Preserving Dimensionality Reduction : A Fuzzy Logic Approach

Nikhil R. Pal[1*], Gautam K. Mandal[2] and Eluri Vijaya Kumar[3]

1. Machine Intelligence Unit, Indian Statistical Institute, Calcutta - 35, INDIA
2. Dept of Physics, Surendranath College, 24/2 M. G. Road, Calcutta - 9, INDIA
3. Vedika International Pvt Ltd, 5B Sarat Bose Road, Calcutta - 20, INDIA

email: nikhil@isical.ernet.in

### Abstract

We propose a fuzzy rule based method for structure preserving dimensionality reduction. This method selects a small representative 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 for projection. Our rule-based system is computationally very efficient compared to Sammon's method and is quite effective to project new points, i.e., it has good predictability.

## 1. Introduction

Feature extraction and dimensionality reduction are two important problems in pattern recognition and exploratory data analysis. Feature analysis can improve generalization ability of classifiers by eliminating harmful features or retaining informative features, and reduce the space and computational requirements associated with analysis of the data.

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. Dimensionality reduction not only helps in the design of a classifier, it also helps in other exploratory data analysis. It can help in both clustering tendency assessment as well as to decide on the number of clusters by looking at the scatterplot of the lower dimensional data.

Feature extraction can be viewed as an implicit or explicit mapping ($\Phi$) from the $p$-dimensional input space to a $q$-dimensional (usually $q \leq p$) output space [1-4]. There are many methods which differ from each other in the characteristics of the mapping function $\Phi$, how $\Phi$ is learned, and what optimization criterion is used. The mapping function can be either *linear* or *nonlinear*.

Recently a large number of artificial neural networks (ANN) and associated learning algorithms have been proposed for feature extraction and multivariate data projection [4]. Although these methods do not necessarily provide new approaches to feature extraction and data projection (from the viewpoint of functionality performed by the networks), they have some advantages over traditional approaches: (i) Most learning algorithms and neural networks are adaptive in nature, thus they are well-suited for many real environments where adaptive systems are required. (ii) For real-time implementation, neural networks provide good architectures which can be easily implemented using current VLSI and optical technologies. (iii) Neural network implementations offer generalization ability for projecting new data. Yet the performance of these data projection networks is not very satisfactory. This has been discussed elsewhere [5].

In this paper, we present a fuzzy rule based scheme for structure preserving dimensionality reduction. The scheme integrates the theory of statistical subsampling, structure preserving characteristic of Sammon's function and the generalization capability of fuzzy rule based reasoning. To the knowledge of the authors no attempt has been made to exploit the power of fuzzy rule based systems for feature extraction/dimensionality reduction. Unlike Sammon's method, the proposed scheme has predictability and can produce lower dimensional data which are coherent with the original data at a much lower computational cost. The scheme has been compared with original Sammon's algorithm. The proposed scheme is much more efficient in terms of computation time and the quality of the projected data is much better than even the neural network implementations [5].

## 2. Sammon's Method

Sammon [2] proposed a simple yet very useful nonlinear projection algorithm that attempts to preserve the structure by finding $n$ points in $q$-dimensional space such that inter-point distances approximate the corresponding inter-point distances in $p$-dimensional space.

Let $X = \{\mathbf{x}_k \mid \mathbf{x}_k = (x_{k1}, x_{k2}, ..., x_{kp}), k = 1, 2, ..., n\}$ be the set of $n$ input vectors and let $Y = \{\mathbf{y}_k \mid \mathbf{y}_k = (y_{k1}, y_{k2}, ..., y_{kq}), k = 1, 2, ..., n\}$ 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 Euclidian distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Sammon suggested looking for $Y$ minimizing the error function $E$

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

Minimization of $E$ is an unconstrained optimization problem in the $nq$ variables $y_{ij}$, $i = 1, 2, ..., n$; $j = 1, 2, ..., q$. Sammon used the method of steepest descent for (approximate) minimization of $E$. Let $\mathbf{y}_i(t)$ to be the estimate of $\mathbf{y}_i$ at the $t$-th iteration, $\forall i$. Then $\mathbf{y}_i(t+1)$ is given by

$$y_{ij}(t+1) = y_{ij}(t) - \alpha \left[ \frac{\partial E(t)}{\partial y_{ij}(t)} / \left| \frac{\partial^2 E(t)}{\partial y_{ij}(t)^2} \right| \right] \qquad (2)$$

where the non-negative scaler constant $\alpha$ (Sammon called it a magic factor and recommended $\alpha \approx 0.3$ or 0.4) is the step size for gradient search.

*With this method we cannot get an explicit mapping function governing the relationship between patterns in p-space and corresponding patterns in q-space. Therefore, it is not possible to project new points. This method also involves a large amount of computation, as every step within an iteration requires the computation of* $\frac{n(n-1)}{2}$ *distances. The algorithm becomes impractical for large n. 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.*

## 3. Proposed Fuzzy Model For Data Projection

Sammon's projection algorithm demands prohibitively large computation for reasonably big data sets. Apart from this 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. If we can identify the relation between input and the projected data set by a set of fuzzy rules then the task of projecting new points becomes a trivial job. We assume that the data set under consideration has been obtained from a time invariant probability distribution. Under this assumption if we extract the rule base from an adequate sample of the data [5] its performance is expected to be practically the same as that of the system identified from the entire data set. Thus, using the concept of statistical subsampling we can reduce the computational overhead of the entire system identification. To summarize the entire process we use the following steps:

1. Select an adequate representative sample $X^{(s)}$.
2. Project the sample $X^{(s)}$ by Sammon's algorithm to generate $Y^{(s)}$.
3. Extract fuzzy rules from $(X^{(s)}, Y^{(s)})$ as described next.

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

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

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

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

and a partition matrix ( hard or fuzzy ). This clustering result can·be used to extract fuzzy rules [6,11]. 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 $\mathbf{v}^x_i$ 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 $\mathbf{v}^y_i$. 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 $||\mathbf{x}_k - \mathbf{v}^x_i||$ is small, $||\mathbf{y}_k - \mathbf{v}^y_i||$ 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 such a case the $i$-th cluster can be translated into a rule of the form:

Mamdani-Assilian (MA) models [10] : If $\mathbf{x}$ is CLOSE to $\mathbf{v}^x_i$ then $\mathbf{y}$ is CLOSE to $\mathbf{v}^y_i$;
Takagi-Sugeno (TS) models [7] : If $\mathbf{x}$ is CLOSE to $\mathbf{v}^x_i$ then $\mathbf{y} = u_i(\mathbf{v}^y_i)$.

Usually the antecedent part, if $\mathbf{x}$ is CLOSE to $\mathbf{v}^x_i$, is written as a conjunction of $p$ atomic clauses: If $x_1$ is CLOSE to $v^x_{i1}$ and $x_2$ is CLOSE to $v^x_{i2}$ .... and $x_p$ is CLOSE to $v^x_{ip}$. The function $u_i(.)$ in the TS case primarily models the behavior of the input-output relation in the neighborhood of $\mathbf{v}^y_i$. Rules can also be generated when $Y$ is clustered, and then the centers $\{\mathbf{v}^x_i\}$ are generated as centroids of associated crisp clusters in $X$. Similarly, when $X$ is clustered the centers $\{\mathbf{v}^y_i\}$ can be generated as centroids of associated crisp 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 input-output space. One of the simple ways to assign a membership value to the input data $\mathbf{x}_i$ is by

$$\mu(\mathbf{x}_i) = \max\left\{\mu(\mathbf{x}_j^*) = \mu\left(\begin{array}{l} \mathbf{x}_j = \mathbf{x}_i \\ \mathbf{y}_j \end{array}\right), \mathbf{x}_j^* \in \mathcal{X}^*\right\}.$$

And then each component of $\mathbf{x}_i \subset \mathcal{R}^p$ is also assigned the same membership value, i.e., $\mu(x_{ij}) = \mu(\mathbf{x}_i), \forall j = 1, 2, ..., p$.

Before we can actually extract the set of rules we need to decide on several issues [11]:

1. Choice of the clustering algorithm. Although there could be many choices we use the fuzzy c-means (FCM) algorithm.[12]

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 $Y$. Each has its advantages and disadvantages. In this study we decided to use $X^*$.

3. Deciding on the number of rules or clusters. Researchers used different cluster validity indices like the Xie-Beni [8] index, Gath-Geva [9] index and so on. Although these validity indices have been used for fuzzy rule extraction, they have been developed for cluster validation without paying attention to the rule based system identification problem. Use of these indices for the present problem is debatable. In our case we have heuristically decided on the number of clusters.

4. Choice of the structure of the rule base, i.e., deciding on whether MA model or TS model. If TS model is used, what would be the structure of the right hand side. The present investigation is restricted to the TS model only. We have considered two forms for the right hand side. The first one is the most simple form of TS model with a constant for the right hand side of each rule - we call this Scheme-1. The second form uses a linear combination of input variables which we call Scheme-2.

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 overall square error on the training data as an index for validation. In the present case, we are considering Sammon's error for validation of the system.

Let $\mathbf{v}_i^*$, $i = 1, 2, ..., c$ be the center of the clusters obtained by FCM on $X^*$. We translate the $i$-th cluster into a rule of the form : $R_i$ : If $\mathbf{x}_k$ is CLOSE to $\mathbf{v}_i^x$ then $\mathbf{y}_k = u_i(\mathbf{v}_i^y)$. Note that "$\mathbf{x}_k$ is CLOSE to $\mathbf{v}_i^x$" is essentially an antecedent clause with $p$ components. Thus, $R_i$ : If $x_{k1}$ is CLOSE to $v_{i1}^x$ ... and $x_{kp}$ is CLOSE to $v_{ip}^x$ then $\mathbf{y}_k = u_i(\mathbf{v}_i^y)$. Therefore, each cluster is translated into 1 rule. (Since $\mathbf{y}_k \in \mathcal{R}^q$, $R_i$ can be viewed as q different rules, one for each component of $\mathbf{y}_k$.) This set of $c$ rules form an initial rule base for data projection. For an input vector $\mathbf{x}_k \in \mathcal{R}^p$ let $\alpha_i$ be the firing strength of the rule $R_i$ computed using any conjunction operator (say product). Then $\widehat{\mathbf{y}}_k = (\widehat{y}_{k1}, \widehat{y}_{k2}, ..., \widehat{y}_{kq})^T$ is computed as

$$\widehat{\mathbf{y}}_k = \frac{\sum_{i=1}^{c} \alpha_i . u_i(\mathbf{v}_i^y)}{\sum_{i=1}^{c} \alpha_i}. \tag{3}$$

## 3.1 Scheme-1

Here, we take the rules $R_i$ = If $x_{k1}$ is CLOSE to $v_{i1}^x$ ... and $x_{kp}$ is CLOSE to $v_{ip}^x$ then $\mathbf{y}_k = \mathbf{v}_i^y$. In order to implement the rule base we need to define the membership function for "$x_{kj}$ CLOSE to $v_{ij}^x$". Here, as an initial choice we used symmetric triangular functions having peak ( a point with membership 1 ), $a_{ij} = v_{ij}^x$ and width $b_{ij}$. Note that $a_{ij}$'s and $b_{ij}$'s for all q rules corresponding to a particular cluster $i$, are the same. For the $j$-th feature, to find $b_{ij}$ we proceed as follows. We sort $v_{ij}^x$, $i = 1, 2, ..., c$. Let the sorted list be $v_{i_lj}^x$, $l = 1, 2, ..., c$. Suppose $v_{ij}^x$ takes the $m$-th position in the sorted list, i.e., $v_{i_mj}^x = v_{ij}^x$, then the width of the fuzzy set associated to $v_{ij}^x$, (i.e., the $m$-th fuzzy set on the axis for $j$-th feature) is

$$\begin{aligned} b_{m,j} &= 2 * Max\{(v_{i_mj}^x - v_{i_{m-1}j}^x), \\ &\quad (v_{i_{m+1}j}^x - v_{i_mj}^x)\}, \quad m = 2, ..., c-1, \qquad (4) \\ b_{1,j} &= 2*Max\{(v_{i_1j}^x - (L_j - (0.05*(H_j - L_j))), \\ &\quad (v_{i_2j}^x - v_{i_1j}^x)\}, \qquad (5) \\ b_{c,j} &= 2 * Max\{(v_{i_cj}^x - v_{i_{c-1}j}^x), \\ &\quad (H_j + (0.05 * (H_j - L_j))) - v_{i_cj}^x)\}. \qquad (6) \end{aligned}$$

Here $L_j$ and $H_j$ are the lowest and highest value of feature $j$. Note that (5) and (6) extend the domain of the $j$-th feature which would be helpful for points not used to train the system. This particular choice of $b_{ij}$'s actually extends the domain of the $j$-th feature by 10%. This is done keeping in view of two things: (1) each cluster corresponds to a local relation in the input-output space so the extended domain should not be much bigger than $[L_j, H_j]$ and (2) the clustering is done on some representative sample so the system may generate points outside $[L_j, H_j]$.

Then $a_{ij}$, $b_{ij}$ and $v_{ij}^y$ are tuned using gradient descent to minimize the error $\sum_{k=1}^{n} ||\widehat{y}_k - y_k||^2$.

## 3.2 Scheme-2

In Scheme-1 since the right hand side of each rule has only a constant value, it may not be adequate for modeling complex data structures. Our computational exercise, indeed reveals this. Hence we consider TS model with consequents as a linear function of the antecedent variables. Here the rules $R_i$ take the form:

$R_i$ = If $\mathbf{x}_k$ is CLOSE to $\mathbf{v}_i^x$ then $y_{kj} = f_{ij}(\mathbf{x}_k)$ , $j = 1, 2, ..., q$.
We use $f_{ij} = d_{ij0} + d_{ij1}.x_{k1} + ... + d_{ijp}.x_{kp}, j = 1, 2, ..., q$;
$d_{ijt}; i = 1, 2, .., c; j = 1, 2, .., q; t = 0, 1, .., p$ are constants to be identified.

In order to implement the rule base we need to define the membership function for "$x_j$ CLOSE to $v_{ij}^x$". Here we use more general *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, $b_{ij}^L$'s and $b_{ij}^R$'s for all q rules corresponding to a particular cluster $i$ are the same. For the $j$-th feature, to find $b_{ij}^L$ and $b_{ij}^R$ we proceed as follows. We sort $v_{ij}^x$, $i = 1, 2, ..., c$. Let the sorted list be $v_{i_l j}^x$, $l = 1, 2, ..., c$. Suppose $v_{ij}^x$ takes the $m$-th 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 $m$-th fuzzy set on the axis for $j$-th feature) is

$$b_{m,\ j}^L = \{v_{i_m j}^x - v_{i_{m-1} j}^x\},\ m = 2, ..., c-1$$

$$b_{m,\ j}^R = \{v_{i_{m+1} j}^x - v_{i_m j}^x\},\ m = 2, ..., c-1$$

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

$$b_1^R = \{v_{i_2 j}^x - v_{i_1 j}^x\}$$

$$b_c^L = \{v_{i_c j}^x - v_{i_{c-1} j}^x\}$$

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

We have obtained the least square error (LSE) estimate of the consequent parameters assuming fixed values for the antecedent parameters. We take this as an initial choice for the consequent. We can now use the gradient descent method to further refine all parameters (memberships and centers). 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. In this report, we don't consider this tuning because we got excellent results even without this.

## 4. Results

To demonstrate the effectiveness of the proposed scheme we implemented Sammon's algorithm also. All algorithms are tested on three data sets named **Iris**, **Sphere-Shell**, and **Elongated-Clusters**. Iris is a well-known data set consisting 150 points from three classes in a 4-dimensional space. Each class has 50 points. One of the classes is well separated from the rest while the other two have some overlap. Sphere-Shell is a synthetic data set consisting of 1000 points in 3-dimension. 500 points are selected randomly within a hemi-sphere of radius r1 and rest 500 in a shell defined by two hemi-spheres of radius r2 and r3, r1 < r2 < r3 . Elongated-Cluster [4] is also a synthetic data set consisting of 2 elongated clusters of 500 points each in 3-space.

We used the following parameter values for the results reported.

**Sammon_Projection** : For all data sets - Error bound =0.0001, Epochs = 200 and Learning rate = 0.4.

**For Scheme-1** : For Iris and Sphere-Shell - Learning rate for width = 0.1, Learning rate for center = 0.1, Learning rate for consequent = 0.45, Epochs = 1000 and Rules = 10. For Elongated-Cluster - Learning rate for width = 0.075, Learning rate for center = 0.0.075, Learning rate for consequent = 0.25, Epochs = 2000 and Rules = 10.

**For Scheme-2** : For all data sets 10 rules are used. For Iris we used 75 data points for extraction of the rules, while for Elongated-Clusters and Sphere-Shell only 25% (i.e., 250) points are used for rule extraction. Of course, the entire data set is then projected using the extracted rules.

Table 1 shows the CPU time needed by different methods for the three data sets. For small data sets like Iris Sammon's method require computation time comparable to the proposed schemes. The quality of the outputs both visually and in terms of Sammon's error computed for the entire data set after projection (Table 2) are quite comparable. Figures 1-5 depict the 2-D scatterplots of the projected points by Sammon's method and the proposed schemes. Note that for Sammon's method we use the entire data set.

Table 1: CPU Time (Secs) For Various Methods

| *Data* | *SM* | *Scheme-1* | *Scheme-2* |
|---|---|---|---|
| IRIS | 145 | 183 | 82 |
| Elongated Clusters | 13414 | 1382 | 838 |
| Sphere Shell | 13419 | 1108 | 838 |

Table 2: Sammon's Error For Various Methods

| *Data* | *SM* | *Scheme-1* | *Scheme-2* |
|---|---|---|---|
| IRIS | 0.006341 | 0.015662 | 0.033050 |
| Elongated Clusters | 0.000710 | 0.023999 | 0.000758 |
| Sphere Shell | 0.05213 | 0.034104 | 0.032565 |

For the Elongated-Clusters proposed Scheme-1 and Scheme -2 reduce the CPU time by about 90% and 94% respectively. Even for such a complex data structure the performance of the proposed schemes and Sammon's method are quite similar in terms of Sammon's error function (Table 1). Visually also they are quite comparable (Figs. 1-5).

Finally, for the Sphere-Shell also a significant improvement (about 90%) in computation time is exhibited by the proposed methods over Sammon's algorithm.

## 5. Conclusions

We have proposed a new fuzzy rule based scheme for structure preserving dimensionality reduction (feature extraction). It is based on statistical theory of subsampling and universal approximation capability of rule-based fuzzy systems. We used exploratory data analysis for extraction of the initial rule set which is then further tuned using gradient descent. We tested the proposed schemes on several data sets and obtained excellent results. Our method achieved three things : (i) Unlike Sammon's method it has good predictability. (ii) Computationally it is also much more efficient than original Sammon's method. (iii) It is more efficient (both in terms of computation time and predictability) than some of the NN implementations of Sammon's method (We have not included results for NN methods due to lack of space). In near future we would like to investigate performance of the proposed scheme with the MA model also.

### References

1. K. Fukunga, Introduction to Statistical Pattern Recognition, Second Edition, New York Academic Press. 1990.
2. J. W. Sammon Jr., "A nonlinear mapping for data structure analysis," IEEE Trans. Computers, vol. C-18, pp. 401-409, 1969.
3. B. Schachter, "A Nonlinear mapping algorithm for large databases," Comput. Graphics Image Process., vol. 7, pp. 271-278, 1978.
4. J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," IEEE Trans. on Neural Networks, vol. 6, No. 2, pp. 296-317, 1995.
5. N. R. Pal and E. V. Kumar, " Neural networks for dimensionality reduction", 4th Int. Conf. on Neural Info. Processing., ICONIP'97, New Zealand, Vol. 1, 221-224, 1997.
6. M. Delgado, A. F. Gomez-Skarmeta and F. Martin, " A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling", IEEE Trans. on Fuzzy Systems, vol. 5, no.2, pp. 223-233, 1997.
7. T. Takagi and M Sugeno, " Fuzzy identification of systems and its application to modeling and control", IEEE Trans. Syst., Man, Cybern., vol. SMC-15, no. 1, pp. 116-132, 1985.
8. X. L. Xie and G. A. Beni, " Validity measure for fuzzy clustering," IEEE Trans. Pattern Anal. Machine Intell., vol 3, n. 8, pp. 841-846, 1991.
9. T. Gath and A. B. Geva, " Unsupervised optimal fuzzy clustering", IEEE Trans. PAMI, 11(7), 773-781, 1989.
10. 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.
11. N. R. Pal , K. Pal, J. C. Bezdek and T. Runkler, " Some issues in system identification using clustering", Proc. Int. Conf. on Neural Networks, ICNN, 1997, IEEE Press, NJ, 2524-2529, 1997.
12. J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, NewYork : Plenum, 1981.

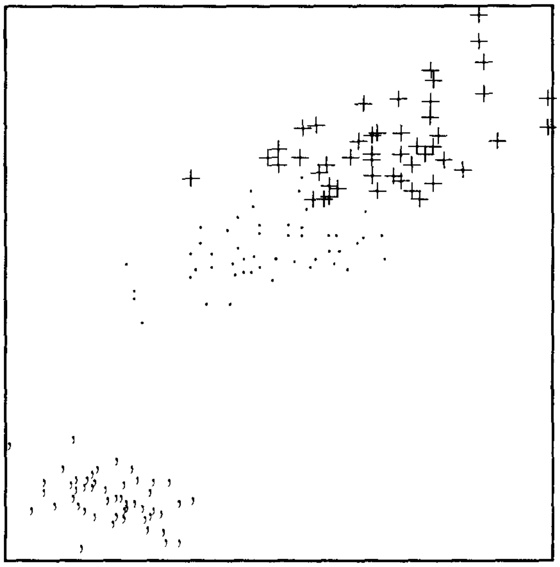

Figure 1: Iris - Sammon's Projection

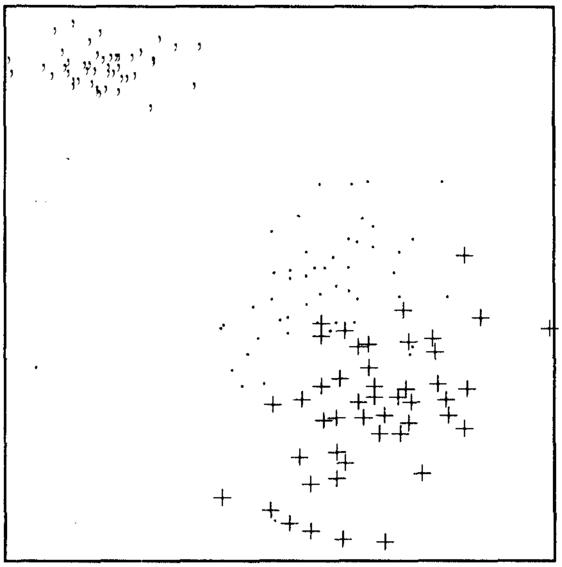

Figure 2: Iris - Scheme-1

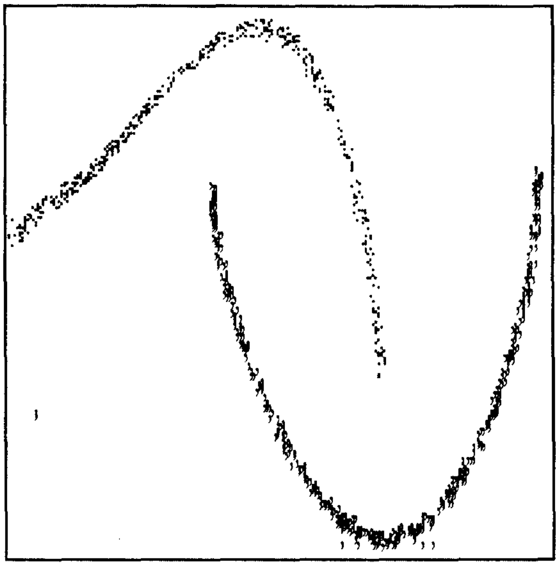

Figure 4: Elongated Clusters - Sammon's Projection

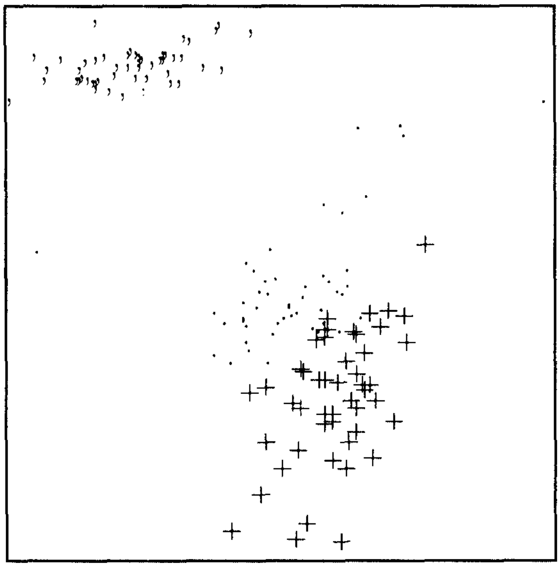

Figure 3: Iris - Scheme-2

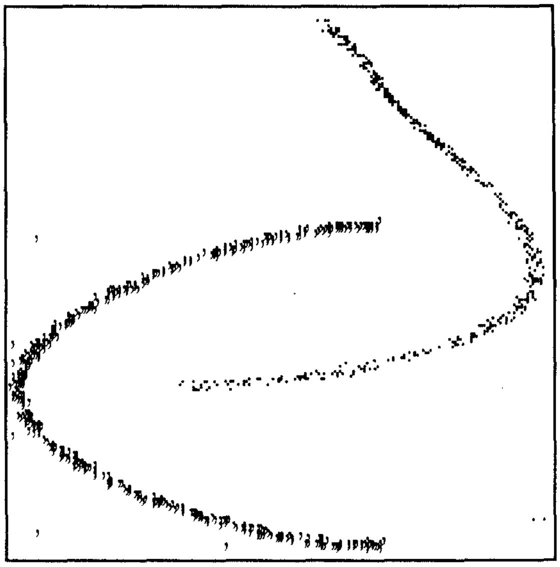

Figure 5: Elongated Clusters - Scheme-2