Fuzzy Clustering and Data Analysis Toolbox

For Use with **Matlab**

Balazs Balasko, Janos Abonyi and Balazs Feil

Preface

About the Toolbox

The Fuzzy Clustering and Data Analysis Toolbox is a collection of MATLAB functions. Its propose is to divide a given data set into subsets (called *clusters*), hard and fuzzy partitioning mean, that these transitions between the subsets are crisp or gradual.
The toolbox provides four categories of functions:

- **Clustering algorithms.** These functions group the given data set into clusters by different approaches: functions Kmeans and Kmedoid are hard partitioning methods, FCMclust, GKclust, GGclust are fuzzy partitioning methods with different distance norms that are defined in Section 1.2.

- **Evaluation with cluster prototypes.** On the score of the clustering results of a data set there is a possibility to calculate membership for "unseen" data sets with these set of functions. In 2-dimensional case the functions draw a contour-map in the data space to visualize the results.

- **Validation.** The validity function provides cluster validity measures for each partition. It is useful, when the number of cluster is unknown *apriori.* The optimal partition can be determined by the point of the extrema of the validation indexes in dependence of the number of clusters. The indexes calculated are:Partition Coefficient(PC), Classification Entropy(CE),Partition Index(SC), Separation Index(S), Xie and Beni's Index(XB), Dunn's Index(DI) and Alternative Dunn Index(DII).

- **Visualization** The Visualization part of this toolbox provides the modified Sammon mapping of the data. This mapping method is a multidimensional scaling method described by Sammon. The original method is computationally expensive when a new data point has to be mapped, so a modified method described by Abonyi got into this toolbox.

- **Example.** An example based on industrial data set to present the usefulness of the purpose of these algorithms.

Installation

The installation is straightforward and it does not require any changes to your system settings. If you would like to use these functions, just copy the directory ”FUZZCLUST” within its files where the directory ”toolbox” is situated (...\ MATLAB\ TOOLBOX \ ...).

Contact

Janos Abonyi or Balazs Feil:
Department of Process Engineering University of Veszprem
P.O.Box 158 H-8200, Veszprem, Hungary
Phone: +36-88-422-022/4209 Fax: +36-88-421-709
E-mail: abonyij@fmt.vein.hu, feilb@fmt.vein.hu
Web: (www.fmt.vein.hu/softcomp)

Contents

Chapter 1

Theoretical introduction

The aim of this chapter is to introduce the theories in the fuzzy clustering so that one can understand the subsequent chapter of this thesis at a necessary level. Fuzzy clustering can be used as a tool to obtain the partitioning of data. Section 1.1 gives the basic notions about the data, clusters and different types of partitioning. Section 1.2 presents the description of the algorithms used in this toolbox. The validation of these algorithms is described in Section 1.3. Each discussed algorithm has a demonstrating example in Chapter 2 at the description of the MATLAB functions.

For a more detailed treatment of this subject see the classical monograph by Bezdek [1] or Hoppner's book [2]. The notations and the descriptions of the algorithms closely follow the structure used by [3].

1.1 Cluster analysis

The objective of cluster analysis is the classification of objects according to similarities among them, and organizing of data into groups. Clustering techniques are among the *unsupervised* methods, they do not use prior class identifiers. The main potential of clustering is to detect the underlying structure in data, not only for classification and pattern recognition, but for model reduction and optimization.

Different classifications can be related to the algorithmic approach of the clustering techniques. *Partitioning, hierarchical, graph-theoretic methods* and methods based on *objective function* can be distinguished. In this work we have worked out a toolbox for the partitioning methods, especially for hard and fuzzy partition methods.

1.1.1 The data

Clustering techniques can be applied to data that is quantitative (numerical), qualitative (categoric), or a mixture of both. In this thesis, the clustering of quantitative data is considered. The data are typically observations of some physical process. Each observation consists of n measured variables, grouped into an n-dimensional row vector $\mathbf{x}_k = [x_{k1}, x_{k2}, \dots, x_{kn}]^T, \mathbf{x}_k \in \mathbf{R}^n$. A set of N observations is denoted by $\mathbf{X} = \{\mathbf{x}_k | k = 1, 2, \dots, N\}$, and is represented as an $N \times n$ matrix:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix}. \tag{1.1}$$

In pattern recognition terminology, the rows of $\mathbf{X}$ are called *patterns* or objects, the columns are called the *features* or attributes, and $\mathbf{X}$ is called the *pattern matrix*. In this thesis, $\mathbf{X}$ is often referred to simply as the *data matrix*.

The meaning of the columns and rows of $\mathbf{X}$ with respect to reality depends on the context. In medical diagnosis, for instance, the rows of $\mathbf{X}$ may represent patients, and the columns are then symptoms, or laboratory measurements for the patients. When clustering is applied to the modeling and identification of dynamic systems, the rows of $\mathbf{X}$ contain samples of time signals, and the columns are, for instance, physical variables observed in the system (position, velocity, temperature, etc.). In order to represent the system's dynamics, past values of the variables are typically included in $\mathbf{X}$ as well.

In system identification, the purpose of clustering is to find relationships between independent system variables, called the regressors, and future values of dependent variables, called the regressands. One should, however, realize that the relations revealed by clustering are just causal associations among the data vectors, and as such do not yet constitute a prediction model of the given system. To obtain such a model, additional steps are needed.

1.1.2 The clusters

Various definitions of a cluster can be formulated, depending on the objective of clustering. Generally, one may accept the view that a cluster is a group of objects that are more similar to one another than to members of other clusters. The term "similarity" should be understood as mathematical similarity, measured in some well-defined sense. In metric spaces, similarity is often defined by means of a *distance norm*. Distance can be measured among the data vectors themselves, or as a distance form a data vector to some prototypical object of the cluster. The prototypes are usually not known beforehand, and are sought by the clustering algorithms simultaneously with the partitioning of the data. The prototypes may be vectors of the same dimension as the data objects, but they can also be defined as "higher-level" geometrical objects, such as linear or nonlinear subspaces or functions.

Data can reveal clusters of different geometrical shapes, sizes and densities as demonstrated in Fig. 1.1. Clusters can be spherical (a), elongated or "linear" (b), and also hollow (c) and (d). Clusters (b) to (d) can be characterized as linear and nonlinear subspaces of the data space ($\mathbf{R}^2$ in this case). Algorithms that can detect subspaces of the data space are of particular interest for identification. The performance of most clustering algorithms is influenced not only by the geometrical shapes and densities of the individual clusters but also by the spatial relations and distances among the clusters. Clusters can be well-separated, continuously connected to each other, or overlapping each other.

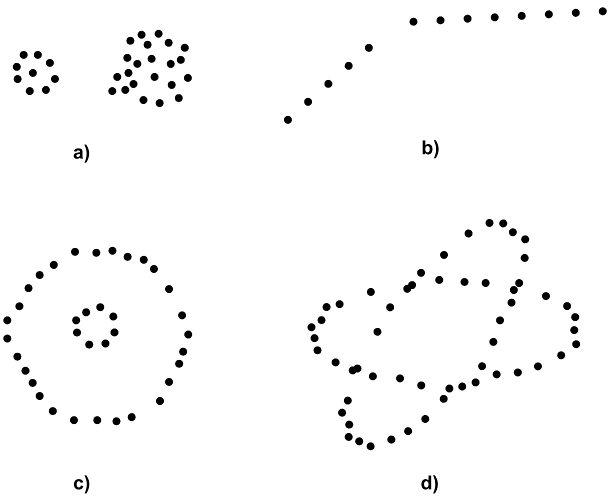

Figure 1.1: Clusters of different shapes and dimensions in $\mathbf{R}^2$.

1.1.3 Cluster partition

Since clusters can formally be seen as subsets of the data set, one possible classification of clustering methods can be according to whether the subsets are fuzzy or crisp (hard). Hard clustering methods are based on classical set theory, and require that an object either does or does not belong to a cluster. Hard clustering in a data set $\mathbf{X}$ means partitioning the data into a specified

number of mutually exclusive subsets of $\mathbf{X}$. The number of subsets (clusters) is denoted by c. Fuzzy clustering methods allow objects to belong to several clusters simultaneously, with different degrees of membership. The data set $\mathbf{X}$ is thus partitioned into c fuzzy subsets. In many real situations, fuzzy clustering is more natural than hard clustering, as objects on the boundaries between several classes are not forced to fully belong to one of the classes, but rather are assigned membership degrees between 0 and 1 indicating their partial memberships. The discrete nature of hard partitioning also causes analytical and algorithmic intractability of algorithms based on analytic functionals, since these functionals are not differentiable.
The structure of the partition matrix $\mathbf{U} = [\mu_{ik}]$:

$$\mathbf{U}{=} \begin{bmatrix} \mu_{1,1} & \mu_{1,2} & \cdots & \mu_{1,c} \\ \mu_{2,1} & \mu_{2,2} & \cdots & \mu_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{N,1} & \mu_{N,2} & \cdots & \mu_{N,c} \end{bmatrix}.$$

Hard partition

The objective of clustering is to partition the data set $\mathbf{X}$ into c clusters. For the time being, assume that c is known, based on prior knowledge, for instance, or it is a trial value, of witch partition results must be validated[1].
Using classical sets, a hard partition can be defined as a family of subsets $\{A_i | 1 \leq i \leq c \subset P(X)\}$, its properties are as follows:

$$\bigcup_{i=1}^{c} A_i = \mathbf{X}, \tag{1.2}$$

$$A_i \cap A_j,\ 1 \leq i \neq j \leq c, \tag{1.3}$$

$$\emptyset \subset A_i \subset \mathbf{X},\ 1 \leq i \leq c. \tag{1.4}$$

These conditions mean that the subsets A_i contain all the data in $\mathbf{X}$, they must be disjoint and none of them is empty nor contains all the data in $\mathbf{X}$. Expressed in the terms of *membership functions*:

$$\bigvee_{i=1}^{c} \mu_{A_i} = 1, \tag{1.5}$$

$$\mu_{A_i} \vee \mu_{A_j},\ 1 \leq i \neq j \leq c, \tag{1.6}$$

$$0 < \mu_{A_i} < 1,\ 1 \leq i \leq c. \tag{1.7}$$

Here μ_{A_i} is the characteristic function of the subset A_i and its value can be zero or one.
To simplify the notations, we use μ_i instead of μ_{A_i}, and denoting $\mu_i(x_k)$ by μ_{ik}, partitions can be represented in a matrix notation.

A $N \times c$ matrix $\mathbf{U} = [\mu_{ik}]$ represents the hard partition if and only if its elements satisfy:

$$\mu_{ij} \in 0, 1, \; 1 \leq i \leq N, \; 1 \leq k \leq c, \tag{1.8}$$

$$\sum_{k=1}^{c} \mu_{ik} = 1, \; 1 \leq i \leq N, \tag{1.9}$$

$$0 < \sum_{i=1}^{N} \mu_{ik} < N, \; 1 \leq k \leq c. \tag{1.10}$$

Definition: Hard partitioning space *Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ be a finite set and let $2 \leq c < N$ be an integer. The hard partitioning space for $\mathbf{X}$ is the set*

$$M_{hc} = \{\mathbf{U} \in \mathbf{R}^{N \times c} | \mu_{ik} \in 0, 1, \forall i, k; \sum_{k=1}^{c} \mu_{ik} = 1, \forall i; 0 < \sum_{i=1}^{N} \mu_{ik} < N, \forall k\}. \tag{1.11}$$

Fuzzy partition

Fuzzy partition can be seen as a generalization of hard partition, it allows μ_{ik} to attain real values in [0,1].
A $N \times c$ matrix $\mathbf{U} = [\mu_{ik}]$ represents the fuzzy partitions, its conditions are given by:

$$\mu_{ij} \in [0, 1], \; 1 \leq i \leq N, \; 1 \leq k \leq c, \tag{1.12}$$

$$\sum_{k=1}^{c} \mu_{ik} = 1, \; 1 \leq i \leq N, \tag{1.13}$$

$$0 < \sum_{i=1}^{N} \mu_{ik} < N, \; 1 \leq k \leq c. \tag{1.14}$$

Definition: Fuzzy partitioning space *Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ be a finite set and let $2 \leq c < N$ be an integer. The fuzzy partitioning space for $\mathbf{X}$ is the set*

$$M_{fc} = \{\mathbf{U} \in \mathbf{R}^{N \times c} | \mu_{ik} \in [0, 1], \forall i, k; \sum_{k=1}^{c} \mu_{ik} = 1, \forall i; 0 < \sum_{i=1}^{N} \mu_{ik} < N, \forall k\}. \tag{1.15}$$

The i-th column of $\mathbf{U}$ contains values of the *membership function* of the i-th fuzzy subset of $\mathbf{X}$.(1.13) constrains the sum of each column to 1, and thus the total membership of each $\mathbf{x}_k$ in $\mathbf{X}$ equals one. The distribution of memberships

among the c fuzzy subsets is not constrained.
We do not discuss the *possibilistic partition* in this work, but we mention, that in case of probabilistic partition the sum of the membership degrees for a data point must not be equal to one .

1.2 Clustering algorithms

1.2.1 K-means and K-medoid algorithms

The hard partitioning methods are simple and popular, though its results are not always reliable and these algorithms have numerical problems as well. From an $N \times n$ dimensional data set K-means and K-medoid algorithms allocates each data point to one of c clusters to minimize the within-cluster sum of squares:

$$\sum_{i=1}^{c} \sum_{k \in A_i} ||\mathbf{x_k} - \mathbf{v_i}||_2 \tag{1.16}$$

where A_i is a set of objects (data points) in the i-th cluster and $\mathbf{v_i}$ is the mean for that points over cluster i. (1.16) denotes actually a distance norm. In K-means clustering $\mathbf{v_i}$ is called the cluster prototypes, i.e. *the cluster centers*:

$$\mathbf{v_i} = \frac{\sum_{k=1}^{N_i} \mathbf{x_k}}{N_i}, \; \mathbf{x_k} \in A_i, \tag{1.17}$$

where N_i is the number of objects in A_i.
In K-medoid clustering the cluster centers are the nearest objects to the mean of data in one cluster $V = \{\mathbf{v_i} \in X | 1 \leq i \leq c\}$. It is useful for example, when each data point denotes a position of a system, so there is no continuity in the data space. In these ways the mean of the points in one set does not exist. The concrete algorithms are described on page 27 and 29 in Chapter 2.

1.2.2 Fuzzy C-means algorithm

The Fuzzy C-means clustering algorithm is based on the minimization of an objective function called *C-means functional*. It is defined by Dunn as:

$$J(\mathbf{X}; \mathbf{U}, \mathbf{V}) = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{ik})^m \|\mathbf{x}_k - \mathbf{v}_i\|_A^2 \tag{1.18}$$

where

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c], \; \mathbf{v}_i \in \mathbf{R}^n \tag{1.19}$$

is a vector of *cluster prototypes* (centers), which have to be determined, and

$$D_{ikA}^2 = \|\mathbf{x}_k - \mathbf{v}_i\|_A^2 = (\mathbf{x}_k - \mathbf{v}_i)^T A(\mathbf{x}_k - \mathbf{v}_i) \tag{1.20}$$

is a squared inner-product distance norm.
Statistically, (1.18) can be seen as a measure of the total variance of $\mathbf{x}_k$ from $\mathbf{v}_i$. The minimization of the c-means functional (1.18) represents a nonlinear optimization problem that can be solved by using a variety of available methods, ranging from grouped coordinate minimization, over simulated annealing to genetic algorithms. The most popular method, however, is a simple Picard iteration through the first-order conditions for stationary points of (1.18), known as the fuzzy c-means (FCM) algorithm.

The stationary points of the objective function (1.18) can be found by adjoining the constraint (1.13) to J by means of Lagrange multipliers:

$$\overline{J}(\mathbf{X}; \mathbf{U}, \mathbf{V}, \lambda) = \sum_{i=1}^{c}\sum_{k=1}^{N}(\mu_{ik})^m D_{ikA}^2 + \sum_{k=1}^{N}\lambda_k\left(\sum_{i=1}^{c}\mu_{ik} - 1\right), \tag{1.21}$$

and by setting the gradients of $(\overline{J})$ with respect to $\mathbf{U}, \mathbf{V}$ and λ to zero. If $D_{ikA}^2 > 0, \forall i, k$ and $m > 1$, then $(\mathbf{U}, \mathbf{V}) \in M_{fc} \times \mathbf{R}^{n\times c}$ may minimize (1.18) only if

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c}\left(D_{ikA}/D_{jkA}\right)^{2/(m-1)}}, \quad 1 \le i \le c,\, 1 \le k \le N, \tag{1.22}$$

and

$$\mathbf{v}_i = \frac{\sum\limits_{k=1}^{N}\mu_{ik}^m \mathbf{x}_k}{\sum\limits_{k=1}^{N}\mu_{i,k}^m}, \quad 1 \le i \le c. \tag{1.23}$$

This solution also satisfies the remaining constraints (1.12) and (1.14). Note that equation (1.23) gives $\mathbf{v}_i$ as the weighted mean of the data items that belong to a cluster, where the weights are the membership degrees. That is why the algorithm is called "c-means". One can see that the FCM algorithm is a simple iteration through (1.22) and (1.23).
The FCM algorithm computes with the standard Euclidean distance norm, which induces hyperspherical clusters. Hence it can only detect clusters with the same shape and orientation, because the common choice of norm inducing matrix is: $\mathbf{A} = \mathbf{I}$ or it can be chosen as an $n \times n$ diagonal matrix that accounts for different variances in the directions in the directions of the coordinate axes of X:

$$\mathbf{A_D} = \begin{bmatrix} (1/\sigma_1)^2 & 0 & \cdots & 0 \\ 0 & (1/\sigma_2)^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & (1/\sigma_n)^2 \end{bmatrix}, \tag{1.24}$$

or A can be defined as the inverse of the $n \times n$ covariance matrix:$A = \mathbf{F}^{-1}$, with

$$\mathbf{F} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{x}_k - \overline{\mathbf{x}})(\mathbf{x}_k - \overline{\mathbf{x}})^T. \tag{1.25}$$

Here $\overline{\mathbf{x}}$ denotes the sample mean of the data. In this case, A induces the Mahalanobis norm on $\mathbf{R}^n$. The concrete algorithm is described on page 33 in Chapter 2.

1.2.3 The Gustafson–Kessel algorithm

Gustafson and Kessel extended the standard fuzzy c-means algorithm by employing an adaptive distance norm, in order to detect clusters of different geometrical shapes in one data set [4]. Each cluster has its own norm-inducing matrix A_i, which yields the following inner-product norm:

$$D^2_{ikA} = (\mathbf{x}_k - \mathbf{v}_i)^T A_i(\mathbf{x}_k - \mathbf{v}_i), \quad 1 \le i \le c, \quad 1 \le k \le N. \tag{1.26}$$

The matrices A_i are used as optimization variables in the c-means functional, thus allowing each cluster to adapt the distance norm to the local topological structure of the data. Let $\mathbf{A}$ denote a c-tuple of the norm-inducing matrices: $\mathbf{A} = (A_1, A_2, ..., A_c)$. The objective functional of the GK algorithm is defined by:

$$J(\mathbf{X}; \mathbf{U}, \mathbf{V}, \mathbf{A}) = \sum_{i=1}^{c}\sum_{k=1}^{N}(\mu_{ik})^m D^2_{ikA_i}. \tag{1.27}$$

For a fixed A, conditions (1.12), (1.13) and (1.14) can be directly applied. However, the objective function (1.27) cannot be directly minimized with respect to A_i, since it is linear in A_i. This means that J can be made as small as desired by simply making A_i less positive definite. To obtain a feasible solution, A_i must be constrained in some way. The usual way of accomplishing this is to constrain the determinant of A_i. Allowing the matrix A_i to vary with its determinant fixed corresponds to optimizing the cluster's shape while its volume remains constant:

$$\|A_i\| = \rho_i, \quad \rho > 0, \tag{1.28}$$

where ρ_i is fixed for each cluster. Using the Lagrange multiplier method, the following expression for A_i is obtained:

$$A_i = [\rho_i det(\boldsymbol{F}_i)]^{1/n}\boldsymbol{F}_i^{-1}, \tag{1.29}$$

where $\boldsymbol{F}_i$ is the *fuzzy covariance matrix* of the i-th cluster defined by:

$$\boldsymbol{F}_i = \frac{\sum_{k=1}^{N}(\mu_{ik})^m(\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T}{\sum_{k=1}^{N}(\mu_{ik})^m}. \tag{1.30}$$

Note that the substitution of (1.29) and (1.30) into (1.26) gives a generalized squared Mahalanobis distance norm between $\mathbf{x}_k$ and the cluster mean $\mathbf{v}_i$, where the covariance is weighted by the membership degrees in $\mathbf{U}$. The concrete algorithm is described on page 37 in Chapter 2.
The numerically robust GK algorithm described by R. Babuška, P.J. van der Veen, and U. Kaymak [5] is used in this toolbox.

1.2.4 The Gath–Geva algorithm

The fuzzy maximum likelihood estimates (FMLE) clustering algorithm employs a distance norm based on the fuzzy maximum likelihood estimates, proposed by Bezdek and Dunn [6]:

$$D_{ik}(\mathbf{x}_k, \mathbf{v}_i) = \frac{\sqrt{\det(\boldsymbol{F}_{wi})}}{\alpha_i} \exp\left(\frac{1}{2}\left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)^T \boldsymbol{F}_{wi}^{-1}\left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)\right) \tag{1.31}$$

Note that, contrary to the GK algorithm, this distance norm involves an exponential term and thus decreases faster than the inner-product norm. $\boldsymbol{F}_{wi}$ denotes the fuzzy covariance matrix of the i-the cluster, given by:

$$\boldsymbol{F}_{wi} = \frac{\sum_{k=1}^{N} (\mu_{ik})^w (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T}{\sum_{k=1}^{N} (\mu_{ik})^w}, \; 1 \le i \le c \tag{1.32}$$

where $w = 1$ in the original FMLE algorithm, but we use the $w = 2$ weighting exponent, so that the partition becomes more fuzzy to compensate the exponential term of the distance norm. The difference between the matrix $\boldsymbol{F}_i$ in GK algoritm and the $\boldsymbol{F}_{wi}$ define above is that the latter does not involve the weighting exponent m, instead of this it consists of $w = 1$. (The reason for using this w exponent is to enable to generalize this expression.) This is because the two weighted covariance matrices arise as generalizations of the classical covariance from two different concepts. The α_i is the prior probability of selecting cluster i, given by:

$$\alpha_i = \frac{1}{N}\sum_{k=1}^{N} \mu_{ik}. \tag{1.33}$$

The membership degrees μ_{ik} are interpreted as the posterior probabilities of selecting the i-th cluster given the data point $\boldsymbol{x}_k$. Gath and Geva [7] reported that the fuzzy maximum likelihood estimates clustering algorithm is able to detect clusters of varying shapes, sizes and densities. The cluster covariance matrix is used in conjunction with an "exponential" distance, and the clusters are not constrained in volume. However, this algorithm is less robust in the sense that it needs a good initialization, since due to the exponential distance norm, it converges to a near local optimum. The concrete algorithm is described on page 41 in Chapter 2.

1.3 Validation

Cluster validity refers to the problem whether a given fuzzy partition fits to the data all. The clustering algorithm always tries to find the best fit for a fixed number of clusters and the parameterized cluster shapes. However this does not mean that even the best fit is meaningful at all. Either the number of clusters might be wrong or the cluster shapes might not correspond to the groups in the data, if the data can be grouped in a meaningful way at all. Two main approaches to determining the appropriate number of clusters in data can be distinguished:

- Starting with a sufficiently large number of clusters, and successively reducing this number by merging clusters that are similar (compatible) with respect to some predefined criteria. This approach is called *compatible cluster merging*.
- Clustering data for different values of c, and using *validity measures* to assess the goodness of the obtained partitions. This can be done in two ways:
  - The first approach is to define a validity function which evaluates a complete partition. An upper bound for the number of clusters must be estimated (c_{max}), and the algorithms have to be run with each $c \in \{2, 3, \ldots, c_{max}\}$. for each partition, the validity function provides a value such that the results of the analysis can be compared indirectly.
  - The second approach consists of the definition of a validity function that evaluates individual clusters of a cluster partition. Again, c_{max} has to be estimated and the cluster analysis has to be carried out for c_{max}. The resulting clusters are compared to each other on the basis of the validity function. Similar clusters are collected in one cluster, very bad clusters are eliminated, so the number of clusters is reduced. The procedure can be repeated until there are *bad* clusters.

Different scalar validity measures have been proposed in the literature, none of them is perfect by oneself, therefor we used several indexes in our Toolbox, which are described below:

1. **Partition Coefficient (PC)**: measures the amount of "overlapping" between cluster. It is defined by Bezdek[1] as follows:

$$PC(c) = \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}(\mu_{ij})^2 \qquad (1.34)$$

where μ_{ij} is the membership of data point j in cluster i.The disadvantage of PC is lack of direct connection to some property of the data themselves. The optimal number of cluster is at the maximum value.

2. **Classification Entropy (CE):**it measures the fuzzyness of the cluster partition only, which is similar to the Partition Coefficient.

$$CE(c) = -\frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{N}\mu_{ij}log(\mu_{ij}), \tag{1.35}$$

3. **Partition Index (SC):** is the ratio of the sum of compactness and separation of the clusters. It is a sum of individual cluster validity measures normalized through division by the fuzzy cardinality of each cluster[8].

$$SC(c) = \sum_{i=1}^{c}\frac{\sum_{j=1}^{N}(\mu_{ij})^m||x_j - v_i||^2}{N_i\sum_{k=1}^{c}||v_k - v_i||^2} \tag{1.36}$$

SC is useful when comparing different partitions having equal number of clusters. A lower value of *SC* indicates a better partition.

4. **Separation Index (S):** on the contrary of partition index (SC), the separation index uses a minimum-distance separation for partition validity[8].

$$S(c) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{N}(\mu_{ij})^2||x_j - v_i||^2}{Nmin_{i,k}||v_k - v_i||^2} \tag{1.37}$$

5. **Xie and Beni's Index (XB):**it aims to quantify the ratio of the total variation within clusters and the separation of clusters[9].

$$XB(c) = \frac{\sum_{i=1}^{c}\sum_{j=1}^{N}(\mu_{ij})^m||x_j - v_i||^2}{Nmin_{i,j}||x_j - v_i||^2} \tag{1.38}$$

The optimal number of clusters should minimize the value of the index.

6. **Dunn's Index (DI):** this index is originally proposed to use at the identification of "compact and well separated clusters". So the result of the clustering has to be recalculated as it was a hard partition algorithm.

$$DI(c) = min_{i\in c}\{min_{j\in c, i\neq j}\{\frac{min_{x\in C_i, y\in C_j}d(x,y)}{max_{k\in c}\{max_{x,y\in C}d(x,y)\}}\}\} \tag{1.39}$$

The main drawback of Dunn's index is computational since calculating becomes computationally very expansive as c and N increase.

7. **Alternative Dunn Index (ADI):**the aim of modifying the original Dunn's index was that the calculation becomes more simple, when the dissimilarity

function between two clusters($min_{x \in C_i, y \in C_j} d(x, y)$) is rated in value from beneath by the triangle-nonequality:

$$d(x, y) \geq |d(y, v_j) - d(x, v_j)| \tag{1.40}$$

where v_j is the cluster center of the j-th cluster.

$$ADI(c) = min_{i \in c}\{min_{j \in c, i \neq j}\{\frac{min_{x_i \in C_i, x_j \in C_j} |d(y, v_j) - d(x_i, v_j)|}{max_{k \in c}\{max_{x,y \in C} d(x, y)\}}\}\} \tag{1.41}$$

Note, that the only difference of *SC, S* and *XB* is the approach of the separation of clusters. In the case of overlapped clusters the values of *DI* and *ADI* are not really reliable because of re-partitioning the results with the hard partition method.

1.4 Visualization

The clustering-based data mining tools are getting popular, since they are able to "learn" the mapping of functions and systems or explore structures and classes in the data.
The Principal Component Analysis maps the data points into a lower dimensional space, which is useful in the analysis and visualization of the correlated high-dimensional data.
We focused on the Sammon mapping method for the visualization of the clustering results,which preserves interpattern distances. This kind of mapping of distances is much closer to the proposition of clustering than simply preserving the variances. Two problems with the Sammon mapping application are:

- The prototypes of clusters are usually not known apriori, and they are calculated along with the partitioning of the data. These prototypes can be vectors dimensionally equal to the examined data points, but they also can be defined as geometrical objects, i.e. linear or non.linear subspaces, functions. Sammon mapping is a projection method, which is based on the preservation of the Euclidian interpoint distance norm, so it can be only used by clustering algorithms calculating with this type of distance norm[10],[11].

- The Sammon mapping algorithm forces to find in a high n-dimensional space N points in a – lower – q-dimensional subspace, such these interpoint distances correspond to the distances measured in the n-dimensional space. This effects a computationally expensive algorithm, since in every iteration step it requires computation of $N(N-1)/2$ distances.

To avoid these problems a modified Sammon mapping algorithm is used in this work, described in Section 1.4.3. The functions PCA and Sammon were taken from SOM Toolbox described by Vesanto[12].

1.4.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. The main objectives of PCA are:

- 1. identify new meaningful underlying variables;
- 2. discover or to reduce the dimensionality of the data set.

The mathematical background lies in "eigen analysis": The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component.

In this work we used the second objective, in that case the covariance matrix of the data set (also called the "data dispersion matrix")is defined as follows:

$$\mathbf{F} = \frac{1}{N} \left(\mathbf{x}_k - \mathbf{v}\right) \left(\mathbf{x}_k - \mathbf{v}\right)^T , \tag{1.42}$$

where $\mathbf{v} = \overline{\mathbf{x}}_\mathbf{k}$, the mean of the data ($N$ equals the number of objects in the data set). Principal Component Analysis (PCA) is based on the projection of correlated high-dimensional data onto a hyperplane[13]. This mapping uses only the first few q nonzero eigenvalues and the corresponding eigenvectors of the $\boldsymbol{F}_i = \boldsymbol{U}_i \Lambda_i \boldsymbol{U}_i^T$, covariance matrix, decomposed to the Λ_i matrix that includes the eigenvalues $\lambda_{i,j}$ of $\boldsymbol{F}_i$ in its diagonal in decreasing order, and to the $\boldsymbol{U}_i$ matrix that includes the eigenvectors corresponding to the eigenvalues in its columns. The vector $\boldsymbol{y}_{i,k} = \boldsymbol{W}_i^{-1} \left(\boldsymbol{x}_k\right) = \boldsymbol{W}_i^T \left(\boldsymbol{x}_k\right)$ is a q-dimensional reduced representation of the observed vector $\boldsymbol{x}_k$, where the $\boldsymbol{W}_i$ weight matrix contains the q principal orthonormal axes in its column $\boldsymbol{W}_i = \boldsymbol{U}_{i,q} \Lambda_{i,q}^{\frac{1}{2}}$.

1.4.2 Sammon mapping

As mentioned, the Sammon mapping method for finding N points in a q-dimensional data space, where the original data are from a higher n-dimensional space. The $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ interpoint distances measured in the n-dimensional space approximate the corresponding $d^*_{ij} = d^*(\mathbf{y}_i, \mathbf{y}_j)$ interpoint distances in the q-dimensional space. This is achieved by minimizing an error criterion, E (called Sammon's stress)[10]:

$$E = \frac{1}{\lambda} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\left(d_{ij} - d^*_{ij}\right)^2}{d_{ij}} \tag{1.43}$$

where $\lambda = \sum_{i<j} d_{ij} = \sum\limits_{i=1}^{N-1} \sum\limits_{j=i+1}^{N} d_{ij}$, but there is no need to maintain λ for a successful solution of the optimization problem, since as a constant, it does not changes the optimization result.

The minimization of E is an optimization problem in the $N * q$ variables $\mathbf{y_{il}}$, $i = 1, 2, \ldots, N$ $l = 1, 2, \ldots, q$, as $\mathbf{y_i} = [y_{i1}, \ldots, y_{iq}]^T$. At the t-th iteration let to be the rating of y_{il} ,

$$y_{il}(t+1) = y_{il}(t) - \alpha \left[\frac{\frac{\partial E(t)}{\partial y_{il}(t)}}{\frac{\partial^2 E(t)}{\partial y_{il}^2(t)}} \right] \tag{1.44}$$

where α is a nonnegative scalar constant (recommended $\alpha \simeq 0.3 - 0.4$), this is the step size for gradient search in the direction of

$$\begin{aligned} \frac{\partial E(t)}{\partial y_{il}(t)} &= -\frac{2}{\lambda} \sum_{k=1, k \neq i}^{N} \left[\frac{d_{ki} - d^*_{ki}}{d_{ki} d^*_{ki}} \right] (y_{il} - y_{kl}) \\ \frac{\partial^2 E(t)}{\partial y_{il}^2(t)} &= -\frac{2}{\lambda} \sum_{k=1, k \neq i}^{N} \frac{1}{d_{ki} d^*_{ki}} \left[(d_{ki} - d^*_{ki}) \right. \\ & \left. - \left(\frac{(y_{il} - y_{kl})^2}{d^*_{ki}} \right) \left(1 + \frac{d_{ki} - d^*_{ki}}{d_{ki}} \right) \right] \end{aligned} \tag{1.45}$$

A drawback of this gradient-descent method is a possibility to reach a local minimum in the error surface, while searching for the minimum of E, so experiments with different random initializations are necessary. The initialization can be estimated based on information which is obtained from the data .

1.4.3 Fuzzy Sammon mapping

Avoiding the drawbacks of Sammon's mapping (see the introduction of this Section) the modified mapping method uses the basic properties of fuzzy clustering algorithms where only the distance between the data points and the cluster centers are considered to be important[14]. The modified algorithm takes into account only $N \times c$ distances, where c represents the number of clusters, weighted by the membership values similarly to (1.18):

$$E_{fuzz} = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{ki})^m \left(d(\mathbf{x}_k, \mathbf{v_i}) - d^*_{ki}\right)^2 \tag{1.46}$$

where $d(\mathbf{x}_k, \mathbf{v_i})$ represents the distance between the $\mathbf{x}_k$ datapoint and the $\mathbf{v_i}$ cluster center measured in the original n-dimensional space, while $d^*_{ki} = d^*(\mathbf{y}_k, \mathbf{z}_i)$ represents the Euclidian distance between the projected cluster center $\mathbf{z}_i$ and the projected data $\mathbf{y}_k$.

This means, in the projected two dimensional space every cluster is represented by a single point, independently to the form of the original cluster prototype, $\mathbf{v_i}$.

The resulted algorithm is similar to the original Sammon mapping, but in this case in every iteration after the adaptation of the projected data points, the projected cluster centers are recalculated based on the weighted mean formula of the fuzzy clustering algorithms described in Chapter 2 on Page 51.

The distances between the projected data and the projected cluster centers are based on the normal Euclidian distance measures. The membership values of the projected data can be also plotted based on the classical formula of the calculation of the membership values:

$$\mu^*_{ki} = \frac{1}{\sum\limits_{j=1}^{c} \left(\frac{d^*(\mathbf{x}_k, \eta_i)}{d^*(\mathbf{x}_k, \mathbf{v_j})}\right)^{\frac{2}{m-1}}} \tag{1.47}$$

and $\mathbf{U}^* = [\mu^*_{ki}]$ is the partition matrix containing the recalculated memberships. The resulted plot will only be an approximation of the original high dimensional clustering in two dimension. The quality of the this rating can be evaluated by determining the maximal value of the mean square error between the original and the re-calculated membership values:

$$P = \|\mathbf{U} - \mathbf{U}^*\| \tag{1.48}$$

Examples are shown in Section 3.3.

Chapter 2

Reference

The Reference chapter provides the descriptions of the functions included into this toolbox. At each description one can find the syntax, the algorithm and two uncomplicated example are considered: a generated disjoint data set in $\mathbf{R^2}$ and a motorcycle real data set: head acceleration of a human "post mortem test object" was plotted in time.
(it can be found on the web: *http://www.ece.pdx.edu/∼mcnames/DataSets/*).
The figures are discussed, and several notes are mentioned as well. The lines of the contour maps mean the level curves of the same values of the membership degree (for details see the description of clusteval on Page 43).
The used input and output arguments are shown in Tab. 2.3 and Tab. 2.4 in the "Function Arguments" section, so one can more easily find out, which function uses which parameter, and what are the output matrix structures. In the following Tab. 2.1 the implemented functions are listed and grouped by proposition.

Table 2.1: **Reference**

	Hard and Fuzzy clustering algorithms	Page
Kmeans	Crisp clustering method with standard Euclidean distance norm	24
Kmedoid	Crisp clustering method with standard standard Euclidean distance norm and centers chosen from the data	28
FCMclust	Fuzzy C-means clustering method with standard Euclidean distance norm	31
GKclust	Fuzzy Gustafson–Kessel clustering method with squared Mahalanobis distance norm	35
GGclust	Fuzzy Gath–Geva clustering method with a distance norm based on the fuzzy maximum likelihood estimates	39
	Evaluation of clustering results	Page
clusteval	calculates the membership values of fuzzy clustering algorithms (C-means, Gustafson-Kessel, Gath-Geva) for ”unseen” data	43
	Validation of the clustering results	Page
validity	Validity measure by calculating different types of validation indexes.	43
	Data normalization	Page
clustnormalize	data normalization with two possible method.	46
clustdenormalize	denormalization of data with the method used through the normalization.	46
	Visualization	Page
PCA	Principal Component Analysis of n-dimensional data.	48
sammon	Sammon Mapping for visualization.	50
FuzSam	Modified Sammon Mapping for visualization of n-dimensional data.	51

Function Arguments

The base syntax of calling a clustering function of this toolbox is:

$$[\mathbf{result}] = function(\mathbf{data}, \mathbf{param}) \tag{2.1}$$

$$[\mathbf{result}] = function(\mathbf{result}, \mathbf{data}, \mathbf{param}) \tag{2.2}$$

This means, the input arguments must be given in structure arrays, as one gets the output matrices in them as well. Some functions use the output matrices as inputs, these are described at their description.

The input data matrix

$data.X$ denotes the matrix of the $N \times n$ dimensional data set X, where N is the number of data points, and n is the number of observations,

$$\mathbf{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,n} \\ X_{1,2} & X_{2,2} & \cdots & X_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{N,1} & X_{N,2} & \cdots & X_{N,n} \end{bmatrix}.$$

The data structures created during normalization are shown in Table 2.2 The input parameters and output matrices are defined in Table 2.3 and Table 2.4.

Table 2.2: **Other data structures**

Argument	Dimension	Description	Used by method
data.Xold	$N \times n$	The original unnormalized data set.	'var', 'range'
data.min	$1 \times n$	The row vector containing the minimum of each data column.	'range'
data.max	$1 \times n$	The row vector containing the maximum of each data column.	'range'
param.mean	$1 \times n$	The row vector containing the mean of each data column.	'var'
data.std	$1 \times n$	The row vector containing the standard deviation of each data column.	'var'

Table 2.3: **The input parameter structures**

Argument	**Notation**	**Description**	**Used by**
param.c	c	The number of clusters or the initial partition matrix. If previous, it must be given as a round scalar with a value greater than one.	kmeans, kmedoid, FCMclust, GKclust GGclust
param.m	m	The weighting exponent which determines the fuzziness of the clusters. It must be given as a scalar greater or equal to one. The default value of m is 2.	FCMclust, GKclust, GGclust
param.e	ϵ	The termination tolerance of the clustering method. The default setting is 0.001.	FCMclust, GKclust, GGclust
param.ro	ρ	The ρ parameter is a row vector, which consists of the determinants i.e. the cluster volumes of the clusters.The default value is one for each cluster.	GKclust
param.vis	-	Visualization parameter for K-means and K-medoid clustering methods. Its value can be zero or one.	kmeans,kmedoid
param.beta	β	A predefined threshold: the maximal ratio between the maximum and minimum eigenvalue of the covariance matrix, with the default value of 10^{15}	GKclust
param.gamma	γ	Weighting parameter, a scaled identity matrix can be added to the covariance matrix in the ratio of γ. Its default value is 0, the maximum is one.	GKclust
param.val	–	It determines, which validity measures will be calculated. It can be chosen 1, 2 or 3.	validity
param.max	–	Maximal number of iteration with a default value of 100.	Sammon, FuzSam
param.alpha	α	Gradient step size, its default value is 0.4.	Sammon, FuzSam

Table 2.4: **The result structures**

Argument	Notation	Dimension	Description	Output of
result.data.f	U	$N \times c$	The structure matrix represents the partition matrix U	Kmeans,Kmedoid, FCMclust, GKclust, GGclust
result.data.d	D_{ik}^2	$N \times c$	The distance matrix containing the square distances between data points and cluster centers.	Kmeans,Kmedoid, FCMclust, GKclust,GGclust
result.cluster.v	V	$c \times n$	Contains the v_i cluster centers.	Kmeans,Kmedoid, FCMclust, GKclust, GGclust
result.cluster.P	P	$n \times n \times c$	Contains the $n \times n$ dimensional covariance matrices for each cluster.	GKclust, GGclust
result.cluster.M	M	$n \times n \times c$	Contains the $n \times n$ dimensional norm-inducing matrices for each cluster.	GKclust
result.cluster.V	V	$c \times n$	Contains the eigenvectors of each cluster.	GKclust, GGclust
result.cluster.D	D	$c \times n$	contains the eigenvalues of each cluster.	GKclust, GGclust
result.cluster.Pi	α_i	$1 \times 1 \times c$	The prior probability for each cluster.	GGclust
result.iter	-	1×1	The number of iteration during the calculation.	Kmeans,Kmedoid, FCMclust, GKclust, GGclust
result.cost	J	$1 \times iter$	The value of the fuzzy C-means functional for each iteration step.	FCMclust, GKclust, GGclust
result.eval.f	U*	$N' \times n$	Partition matrix for the evaluated data set.	clusteval
result.eval.d	D_{ik}^{2*}	$N' \times c$	Distance matrix with the distances between the evaluated data points and the cluster centers.	clusteval
result.proj.P	Y	$N \times q$	Projected data matrix.	PCA, Sammon, FuzSam
result.proj.vp	Z	$c \times q$	Matrix of the projected cluster centers.	PCA, Sammon, FuzSam
result.proj.e	E	$c \times q$	Value of Sammon's stress.	samstr

Kmeans

Purpose

Hard partition of the given data matrix, described inSection 1.2.1.

Syntax

`[result]=Kmeans(data,param)`

Description

The objective function Kmeans algorithm is to partition the data set X into c clusters. Calling the algorithm with the syntax above, first the $param.c$ and $data.X$ must be given, and the output matrices of the function are saved in $result.data.f$, $result.data.d$ and $result.cluster.v$. The number of iteration and the cost function are also saved in the result structure. The calculated cluster center $\mathbf{v_i}$ $(i \in \{1, 2, \ldots, c\})$ is the mean of the data points in cluster i. The difference from the latter discussed algorithms are that it generates random cluster centers, not partition matrix for initialization, so $param.c$ can be given as an initializing matrix containing the cluster centers too.

Example

The examples were generated with the *Kmeanscall* function located in *..\Demos\clusteringexamples\synthetic\* and *..\Demos\clusteringexamples\motorcycle\* directories, where the *nDexample* function and *motorcycle.txt* are situated.

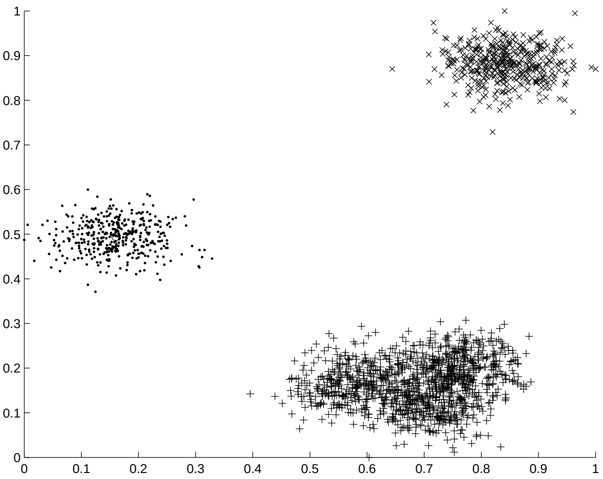

Figure 2.1: Result of K-means clustering by the synthetic disjoint data

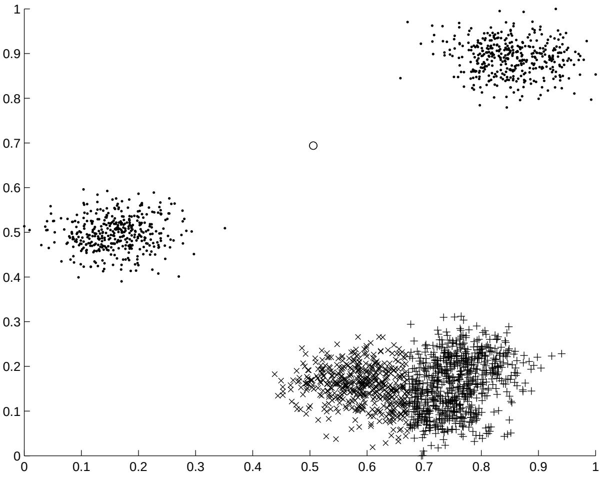

Figure 2.2: Another clustering result of K-means by the synthetic disjoint data

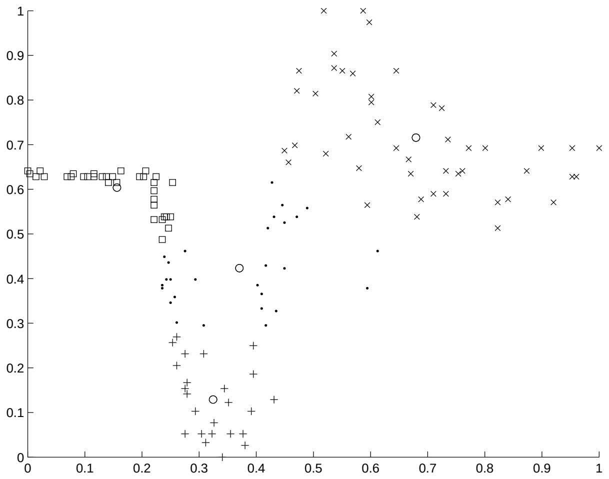

Figure 2.3: The results of the K-means clustering algorithm by the motorcycle data set.

Discussion

In Fig. 2.1 and Fig. 2.2 one can see two results on the synthetic data set with different initializations.(on the latter figure Kmeans was called with $\mathbf{v} = [0.6\ 0.5;\ 0.6\ 0.5;\ 0.6\ 0.5;])$ The difference is obvious. Fig. 2.3 shows the K-means clustered motorcycle data. The cluster centers are marked with 'o'.

Being a hard partition the clustered data points can easily separated by using different markers. If the $param.c$ is greater than 3, the algorithm draws the "border" of the clusters using the Matlab voronoi function.

The main problem of K-means algorithm is that the random initialization of centers, because the calculation can run into wrong results, if the centers "have no data points".

Notes

1. It is recommended to run Kmeans several times to achieve the correct results.

2. To avoid the problem described above, the cluster centers are initialized with randomly chosen data points.

3. If D_{ik} becomes zero for some $\mathbf{x_k}$, singularity occurs in the algorithm, so the initializing centers are not exactly the random data points, they are just near them. (with a distance of 10^{-10} in each dimension)

4. If the initialization problem still occurs for some reason (e.g. the user adds wrong initialization of the function), the "lonely" centers are redefined to data points.

Algorithm

[K-means algorithm] For corresponding theory see Section 1.2.1.
Given the data set $\mathbf{X}$, choose the number of clusters $1 < c < N$.
Initialize with random cluster centers chosen from the data set.

Repeat for $l = 1, 2, \ldots$

Step 1 Compute the distances

$$D_{ik}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T(\mathbf{x}_k - \mathbf{v}_i), \quad 1 \leq i \leq c, \quad 1 \leq k \leq N. \tag{2.3}$$

Step 2 Select the points for a cluster with the minimal distances, they belong to that cluster.

Step 3 Calculate cluster centers

$$v_i^{(l)} = \frac{\sum_{j=1}^{N_i} x_i}{N_i} \tag{2.4}$$

until $\prod_{k=1}^{n} max|v^{(l)} - v^{(l-1)}| \neq 0.$

Ending Calculate the partition matrix

See Also

`Kmedoid, clusteval, validity`

Kmedoid

Purpose

Hard partition of the given data matrix, where cluster centers must be data points (described inSection 1.2.1).

Syntax

[result]=`Kmedoid(`**data**`,`**param**`)`

Description

The objective function Kmedoid algorithm is to partition the data set X into c clusters. The input and output arguments are the same what K-means uses. The main difference between Kmeans and Kmedoid stands in calculating the cluster centers: the new cluster center is the nearest data point to the mean of the cluster points.

The function generates random cluster centers, not partition matrix for initialization, so $param.c$ can be given as an initializing matrix containing the cluster centers too, not only a scalar (the number of clusters)

Example

The examples were generated with the *Kmedoidcall* function located in *..\Demos\clusteringexamples\synthetic\* and *..\Demos\clusteringexamples\motorcycle\* directories, where the *nDexample* function and *motorcycle.txt* are situated.

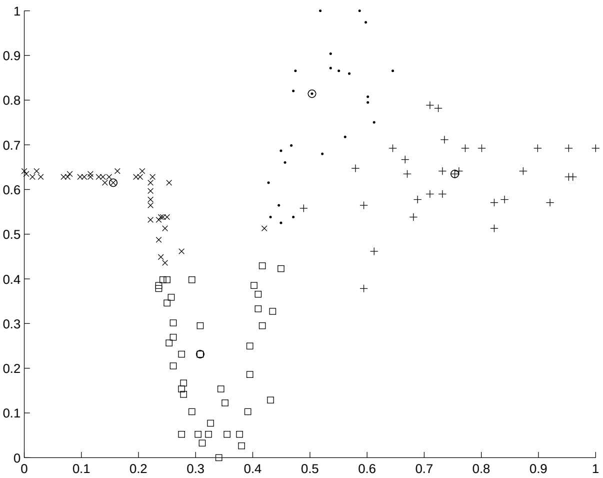

Figure 2.4: The results of the K-medoid clustering algorithm by the motorcycle data set.

Discussion

Different initializations of the algorithm can also effect very different results as Fig. 2.1 and Fig. 2.2 shows. Fig. 2.4 shows the K-medoid clustered motorcycle data. The cluster centers are selected from the data set. Otherwise K-medoid is just like the K-means algorithm.

Notes

See the description of K-means algorithm on Page 27.

Algorithm

[K-medoid algorithm] For corresponding theory see Section 1.2.1.

Given the data set $\mathbf{X}$, choose the number of clusters $1 < c < N$.Initialize with random cluster centers chosen from the data set X.

Repeat for $l = 1, 2, \dots$

Step 1 Compute the distances

$$D_{ik}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T(\mathbf{x}_k - \mathbf{v}_i), \quad 1 \leq i \leq c, \quad 1 \leq k \leq N. \quad (2.5)$$

Step 2 Select the points for a cluster with the minimal distances, they belong to that cluster.

Step 3 Calculate fake cluster centers (the original K-means)

$$v_i^{(l)*} = \frac{\sum_{j=1}^{N_i} x_i}{N_i} \quad (2.6)$$

Step 4 Choose the nearest data point to be the cluster center

$$D_{ik}^{2*} = (\mathbf{x}_k - \mathbf{v^*}_i)^T (\mathbf{x}_k - \mathbf{v^*}_i), \tag{2.7}$$

and

$$x_i^* = argmin_i (D_{ik}^{2*}) \ ; \ v_i^{(l)} = x_i^*. \tag{2.8}$$

until $\prod_{k=1}^{n} max|\mathbf{v}^{(l)} - \mathbf{v}^{(l-1)}| \neq 0.$

Ending Calculate the partition matrix

See Also

`Kmeans, clusteval, validity`

FCMclust

Purpose

Fuzzy C-means clustering of a given data set (described in Section 1.2.2).

Syntax

`[`**result**`]=FCMclust(`**data**`,`**param**`)`

Description

The Fuzzy C-means clustering algorithm uses the minimization of the fuzzy C-means functional (1.18). There are three input parameter needed to run this function: $param.c$, as the number of clusters or initializing partition matrix; $param.m$, as the fuzziness weighting exponent; and $param.e$, as the maximum termination tolerance. The two latter parameter have their default value, if they are not given by the user.
The function calculates with the standard Euclidean distance norm, the norm inducing matrix is an $n \times n$ identity matrix. The result of the partition is collected in structure arrays. One can get the partition matrix cluster centers, the square distances, the number of iteration and the values of the C-means functional at each iteration step.

Example

The examples were generated with the *FCMcall* function located in ..\Demos\clusteringexamples\synthetic\\ and ..\Demos\clusteringexamples\motorcycle\\ directories, where the *nDexample* function and *motorcycle.txt* are situated.

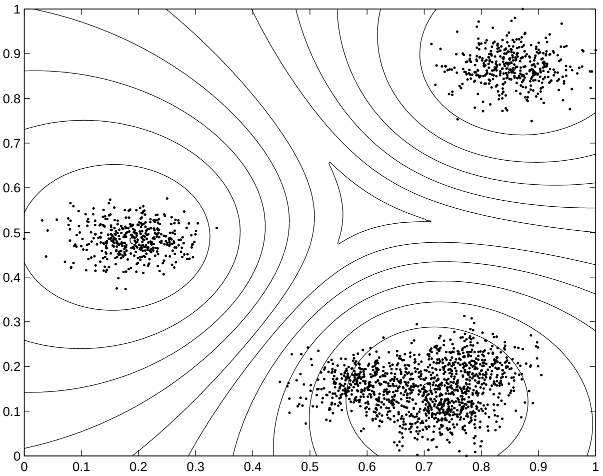

Figure 2.5: The results of the Fuzzy C-means clustering algorithm by a synthetic data set

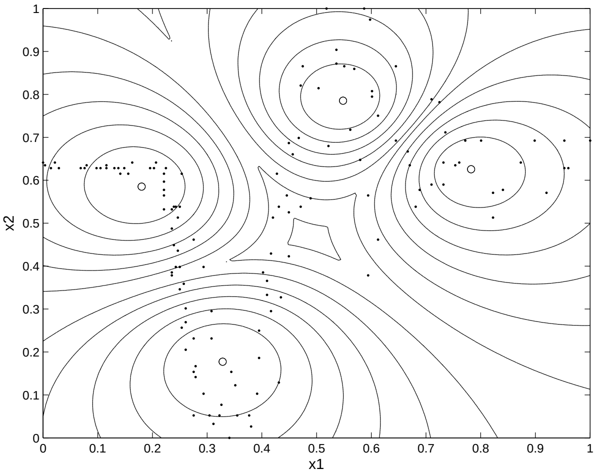

Figure 2.6: The results of the Fuzzy C-means clustering algorithm by the motorcycle data set.

Discussion

In Fig. 2.5 and Fig. 2.6 the '.' remark the data points, the 'o' the cluster centers, which are the weighted mean of the data. The algorithm can only detect clusters with circle shape, that is why it cannot really discover the orientation and shape of the cluster "right below" in Fig. 2.5. In Fig. 2.6 the circles in the contour-map are a little elongated, since the clusters have effect on each other. However the Fuzzy C-means algorithm is a very good initialization tool for more sensitive methods (e.g. Gath-Geva algorithm on page 39).

Notes

1. If D_{ikA} becomes zero for some x_k, singularity occurs in the algorithm: the membership degree cannot be computed.
2. The correct choice of the weighting parameter (m) is important: as m approaches one from above, the partition becomes hard, if it approaches to infinity, the partition becomes maximally fuzzy, i.e. $\mu_{ik} = 1/c$.
3. There is no possibility to use different $\mathbf{A}A$ for the clusters, although in most cases it would be needed.

Algorithm

[FCM algorithm] For corresponding theory see Section 1.2.2.

Given the data set $\mathbf{X}$, choose the number of clusters $1 < c < N$, the weighting exponent $m > 1$, the termination tolerance $\epsilon > 0$ and the norm-inducing matrix $\mathbf{A}$. Initialize the partition matrix randomly, such that $\mathbf{U}^{(0)} \in M_{fc}$.

Repeat for $l = 1, 2, \ldots$

Step 1 Compute the cluster prototypes (means):

$$\mathbf{v}_i^{(l)} = \frac{\sum\limits_{k=1}^{N} (\mu_{ik}^{(l-1)})^m \mathbf{x}_k}{\sum\limits_{k=1}^{N} (\mu_{i,k}^{(l-1)})^m}, \, 1 \leq i \leq c. \tag{2.9}$$

Step 2 Compute the distances:

$$D_{ikA}^2 = (\mathbf{x}_k - \mathbf{v}_i)^T A (\mathbf{x}_k - \mathbf{v}_i), \quad 1 \leq i \leq c, \quad 1 \leq k \leq N. \tag{2.10}$$

Step 3 Update the partition matrix:

$$\mu_{i,k}^{(l)} = \frac{1}{\sum_{j=1}^{c} (D_{ikA}/D_{jkA})^{2/(m-1)}}. \tag{2.11}$$

```
    until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.
```

See Also

`Kmeans, GKclust, GGclust`

References

J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, 1981

GKclust

Purpose

Gustafson-Kessel clustering algorithm extends the Fuzzy C-means algorithm by employing an adaptive distance norm, in order to detect clusters with different geometrical shapes in the data set.

Syntax

`[`**result**`]=GKclust(`**data**`,`**param**`)`

Description

The GKclust function forces, that each cluster has its own norm inducing matrix $\mathbf{A_i}$, so they are allowed to adapt the distance norm to the local topological structure of the data points. The algorithm uses the Mahalanobis distance norm. The parameters are extended with $param.ro$, it is set to one for each cluster by default value.

There are two numerical problems with GK algorithm, which are described in Notes.

Example

The examples were generated with the *GKcall* function located in ..\Demos\clusteringexamples\synthetic\\ and ..\Demos\clusteringexamples\motorcycle\\ directories, where the *nDexample* function and *motorcycle.txt* are situated.

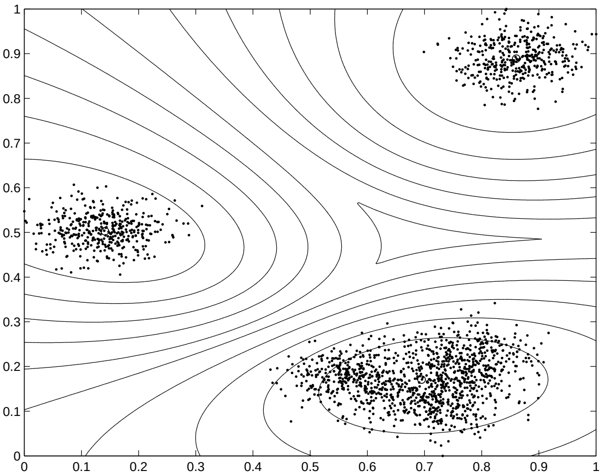

Figure 2.7: The results of the Gustafson-Kessel clustering algorithm by a synthetic data set

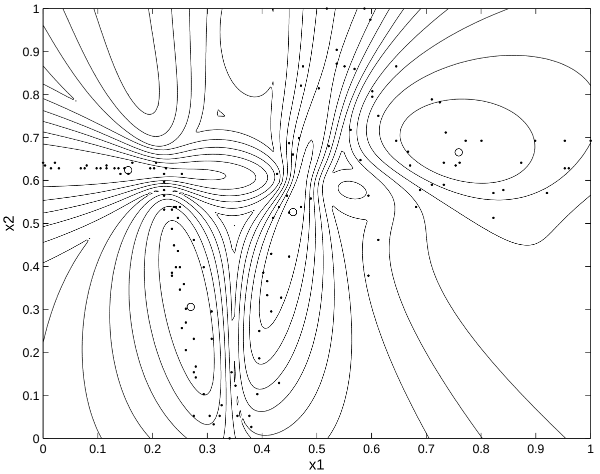

Figure 2.8: The results of the Gustafson-Kessel clustering algorithm by the motorcycle data set.

Discussion

In Fig. 2.7 and Fig. 2.8 the '.' remark the data points, the 'o' the cluster centers. Since this algorithm is an extension of the C-means algorithm (uses adaptive distance norm), it detects the elongated clusters. The orientation and shape can be "mined" from the eigenstructure of the covariance matrix: the direction of the axes are given by the eigenvectors. In Fig. 2.8 the contour-map shows the superposition of the four ellipsoidal clusters.

Notes

1. If there is no prior knowledge, ρ_i is 1 for each cluster, so the GK algorithm can find only clusters with approximately equal volumes.
2. A numerical drawback of GK is: When an eigenvalue is zero or when the ratio between the maximal and the minimal eigenvalue, i.e. the condition number of the covariance matrix is very large, the matrix is nearly singular. Also the normalization to a fixed volume fails, as the determinant becomes zero. In this case it is useful to constrain the ratio between the maximal and minimal eigenvalue, this ratio should be smaller than some predefined threshold, that is in the β parameter.
3. In case of clusters extremely extended in the direction of the largest eigenvalues the computed covariance matrix cannot estimate the underlaying data distribution, so a scaled identity matrix can be added to the covariance matrix by changing the value of γ from zero (as default)to a scalar: $0 \leq \gamma \leq 1$.

Algorithm

[Modified Gustafson-Kessel algorithm] For corresponding theory see Section 1.2.3.

Given the data set $\mathbf{X}$, choose the number of clusters $1 < c < N$, the weighting exponent $m > 1$, the termination tolerance $\epsilon > 0$ and the norm-inducing matrix $\mathbf{A}$. Initialize the partition matrix randomly, such that $\mathbf{U}^{(0)} \in M_{fc}$.

Repeat for $l = 1, 2, \ldots$

Step 1 Calculate the cluster centers.

$$\mathbf{v}_i^{(l)} = \frac{\sum\limits_{k=1}^{N} (\mu_{ik}^{(l-1)})^m \mathbf{x}_k}{\sum\limits_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, 1 \leq i \leq c \qquad (2.12)$$

Step 2 Compute the cluster covariance matrices.

$$\boldsymbol{F}_i^{(l)} = \frac{\sum\limits_{k=1}^{N} (\mu_{ik}^{(l-1)})^m \left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)\left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)^T}{\sum\limits_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, 1 \le i \le c \tag{2.13}$$

Add a scaled identity matrix:

$$\boldsymbol{F}_i := (1-\gamma)\boldsymbol{F}_i + \gamma(\boldsymbol{F}_0)^{1/n}\boldsymbol{I} \tag{2.14}$$

Extract eigenvalues λ_{ij} and eigenvectors ϕ_{ij},
find $\lambda_{i,max} = max_j \lambda_{ij}$ and set:

$$\lambda_{i,max} = \lambda_{ij}/\beta, \quad \forall j \quad for\ which \quad \lambda_{i,max}/\lambda_{ij} \ge \beta \tag{2.15}$$

Reconstruct $\boldsymbol{F}_i$ by:

$$\boldsymbol{F}_i = [\phi_{i,1} \dots \phi_{i,n}] diag(\lambda_{i,1} \dots \lambda_{i,n}) [\phi_{i,1} \dots \phi_{i,n}]^{-1} \tag{2.16}$$

Step 3 Compute the distances.

$$D_{ikA_i}^2(\mathbf{x}_k, \mathbf{v}_i) = \left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)^T \left[(\rho_i det(F_i))^{1/n} F_i^{-1}\right] \left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right) \tag{2.17}$$

Step 4 Update the partition matrix

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{c} \left(D_{ikA_i}(\mathbf{x}_k, \mathbf{v}_i)/D_{jk}(\mathbf{x}_k, \mathbf{v}_j)\right)^{2/(m-1)}}, \quad 1 \le i \le c, 1 \le k \le N\,. \tag{2.18}$$

until $||\boldsymbol{U}^{(l)} - \boldsymbol{U}^{(l-1)}|| < \epsilon$.

See Also

`Kmeans, FCMclust, GGclust`

References

R. Babuska, P.J. van der Veen, and U. Kaymak. Improved covariance estimation for GustafsonKessel clustering. *In Proceedings of 2002 IEEE International Conference on Fuzzy Systems*, pages 10811085, Honolulu, Hawaii, May 2002.

GGclust

Purpose

Gath-Geva clustering algorithm uses a distance norm based on the fuzzy maximum likelihood estimates.

Syntax

`[`**result**`]=GGclust(`**data**`,`**param**`)`

Description

The Gath-Geva clustering algorithm has the same outputs defined at the description of Kmeans and GKclust, but it has less input parameters (only the weighting exponenet and the termination tolerance), because the used distance norm involving the exponential term cannot run into numerical problems.

Note that the original fuzzy maximum likelihood estimates does not involve the value of $param.m$, it is constant 1.

The parameter $param.c$ can be given as an initializing partition matrix or as the number of clusters. For other attributes of the function see Notes.

Example

The examples were generated with the *Kmeanscall* function located in *..\Demos\clusteringexamples\synthetic\* and *..\Demos\clusteringexamples\motorcycle\* directories, where the *nDexample* function and *motorcycle.txt* are situated.

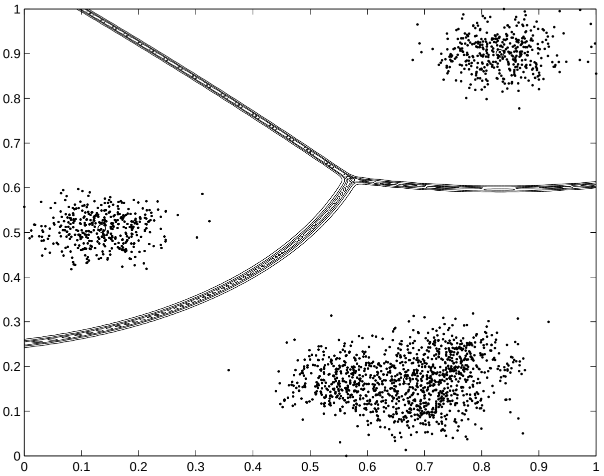

Figure 2.9: The results of the Gath-Geva clustering algorithm by a synthetic data set

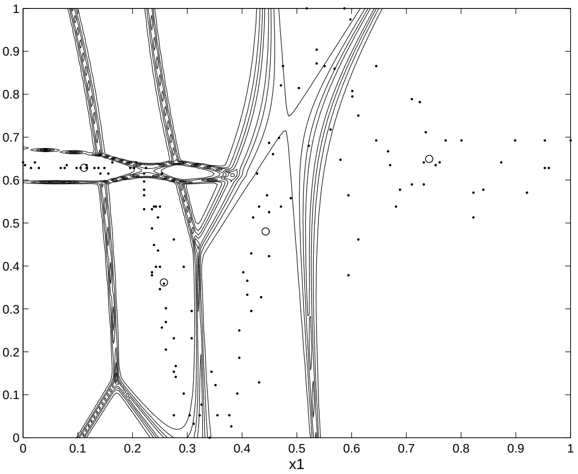

Figure 2.10: The results of the Gath-Geva clustering algorithm by the motorcycle data set.

Discussion

In the figures the '.' remark the data points, the 'o' the cluster centers. Cause of the exponential term in the distance norm (1.31), which decreases faster by increasing $|x_k - v_i|$ distance, the GG algorithm divides the data space into disjoint subspaces shown in Fig. 2.9. In Fig. 2.10 the effects of the other "cluster-borders" distort this disjointness, but the clusters can be easily figured out.

Notes

1. The GG algorithm can overflow at large c values (which indicate small distances) because of invertation problems in (1.31).
2. The fuzzy maximum likelihood estimates clustering algorithm is able to detect clusters of varying shapes, sizes and densities.
3. The cluster covariance matrix is used in conjunction with an "exponential" distance, and the clusters are not constrained in volume.
4. This algorithm is less robust in the sense that it needs a good initialization, since due to the exponential distance norm, it converges to a near local optimum. So it is recommended to use the resulting partition matrix of e.g. FCM to initialize this algorithm.

Algorithm

[Gath-Geva algorithm] For corresponding theory see Section 1.2.4.

Given a set of data $\boldsymbol{X}$ specify c, choose a weighting exponent $m > 1$ and a termination tolerance $\epsilon > 0$. Initialize the partition matrix with a more robust method.

Repeat for $l = 1, 2, \ldots$

Step 1 Calculate the cluster centers:

$$\mathbf{v}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^w \mathbf{x}_k}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^w}, \; 1 \le i \le c$$

Step 2 Compute the distance measure D_{ik}^2.

The distance to the prototype is calculated based the fuzzy covariance matrices of the cluster

$$\boldsymbol{F}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^w \left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right) \left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)^T}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^w}, \; 1 \le i \le c \tag{2.19}$$

The distance function is chosen as

$$D_{ik}^{2}(\mathbf{x}_k, \mathbf{v}_i) = \frac{(2\pi)^{\left(\frac{n}{2}\right)}\sqrt{\mathsf{det}(F_i)}}{\alpha_i}\mathsf{exp}\left(\frac{1}{2}\left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)^T \boldsymbol{F}_i^{-1}\left(\mathbf{x}_k - \mathbf{v}_i^{(l)}\right)\right) \tag{2.20}$$

with the $a\ priori$ probability

$\alpha_i = \frac{1}{N}\sum_{k=1}^{N}\mu_{ik}$

Step 3 Update the partition matrix

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{c}\left(D_{ik}(\mathbf{x}_k, \mathbf{v}_i)/D_{jk}(\mathbf{x}_k, \mathbf{v}_j)\right)^{2/(m-1)}}, \quad 1 \leq i \leq c, 1 \leq k \leq N\,. \tag{2.21}$$

until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$.

See Also

`Kmeans, FCMclust, GKclust`

References

I. Gath and A.B. Geva, Unsupervised Optimal Fuzzy Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:773–781, 1989

clusteval

Purpose

The purpose of the function is to evaluate "unseen" data with the cluster centers calculated by the selected function of clustering method.

Syntax

```
[eval] = clusteval(new,result,param)
```

Description

The clusteval function uses the results and the parameters of FCMclust, GKclust, GGclust clustering functions: It recognizes the used clustering algorithm to evaluate the unseen data sets, on the grounds of the results of clustering.

The new data points to be evaluated must be given in the $N' \times n$ structure array $new.X$, i.e. only the dimensions must be equal for both data sets. The results are collected in $eval.f$ and $eval.d$, as the partition matrix and the distances from cluster prototypes of this data set.

In *2-dimensional case* it generates pair-coordinate points in the data space, calculates the partition matrix, and draws a contour-map by selecting the points with the same partitioning values and drawing default number of colored lines in the data field (the density,i.e. number of lines can be changed with the parameter of the contour function). If the user wants to see this contour-map for a simple clustering, $new.X = data.X$ should be used. The lines of the contour map denotes the position of points with equal membership degrees.

Algorithm

The clusteval function uses the algorithms of the clustering functions, i.e. it can be comprehended as a clustering method with only one iteration step.

Example

The example was generated with the *evalexample* function located in *..\Demos\clustevalexample\* directory, where *data2.txt* is situated.

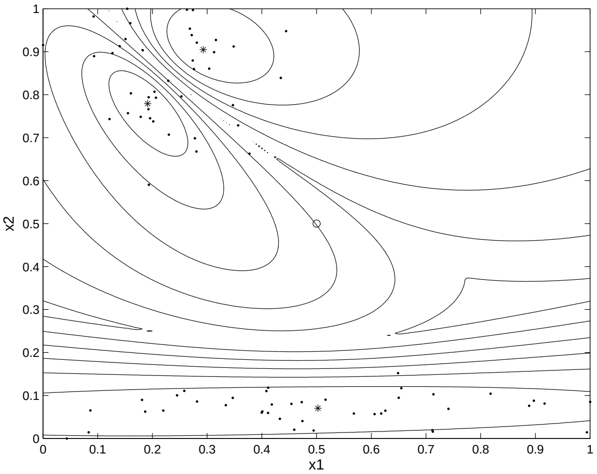

Figure 2.11: The results of the Gustafson-Kessel clustering algorithm.

Discussion

To present the evaluation, a 2-D data set (data2.txt) was selected and the GK-clustering was executed. During the first running $new.X$was equal to $data.X$, so the contour map could be plotted (see Fig. 2.11). After that a new data point was chosen to be evaluated: $new.X = [0.5 \quad 0.5]$ and the result is in $eval.f$.

$$\mathbf{U}_{eval} = [0.0406 \quad 0.5971 \quad 0.3623], \tag{2.22}$$

As Fig. 2.11 also shows the new data point rather belongs to the clusters in the "upper left" corner of the normalized data space.

See Also

FCMclust, GKclust, GGclust

validity

Purpose

Calculating validity measure indexes to estimate the goodness of an algorithm or to help to find the optimal number of clusters for a data set.

Syntax

```
[result] = validity(result,data,param)
```

Description

Depending on the value of $param.val$ the validity function calculates different cluster validity measures.

- $param.val = 1$: the function calculates Partition Coefficient(PC) and Classification Entropy(CE),
- $param.val = 2$: the function calculates Partition Index(SC), Separation Index(S) and Xie and Beni's Index(XB),
- $param.val = 3$: the function calculates Dunn's Index(DI) and Alternative Dunn Index(ADI).

If this parameter is not given by the user or it has another value, the program calculates only PC and CE, as default validity measures. For validation of hard partitioning methods it is recommended to calculate DI and ADI.
The calculation of these indexes is described in Section 1.3.

Example

Examples of the validation indexes are shown in the experimental chapter in Section 3.1 and Section 3.2.

References

J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, 1981, NY.
A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, R.F. Murtagh, Validity-guided (re)clustering with applications to image segmentation, *IEEE Transactions on on Fuzzy Systems*, 4:112 -123, 1996.
X.L. Xie and G.A. Beni, Validity measure for fuzzy clustering, *IEEE Trans. PAMI*, 3(8):841–846, 1991.

clustnormalize and clustdenormalize

Purpose

Normalization and denormalization of the wanted data set.

Syntax

`[data] = clustnormalize(data,method)`

Syntax

`[data] = clustdenormalize(data,method)`

Description

clustnormalize uses two method to data normalization. If method is:
range - Values are normalized between [0,1] (linear operation).
var - Variance is normalized to one (linear operation).
The original data is saved in $data.Xold$, and the function also saves:
1. in case of 'range' the row vectors containing the minimum and the maximum elements of each column from the original data ($data.min$ and $data.max$)
2. in case of 'var' the row vectors containing the mean and standard deviation of the data ($data.mean$ and $data.std$)

clustdenormalize is the inverse of the normalization function: it recognizes, which method was used during the normalization and forms it back.

Example

The example was generated with the *normexample* function located in ..\Demos\normexample\\ directory, where *data3.txt* is situated.

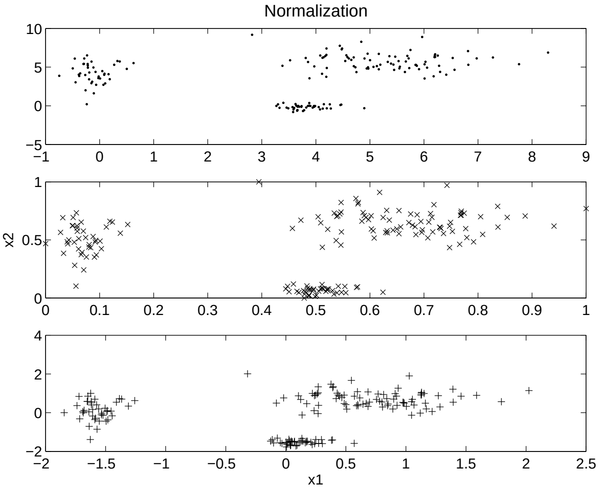

Figure 2.12: Original data, normalized data with 'range' method, normalized data with 'var' method.

Algorithm

Normalization

'range'

$$\mathbf{X} = \frac{\mathbf{X} - \mathbf{X_{min}}}{\mathbf{X_{max}} - \mathbf{X_{min}}} \tag{2.23}$$

'var'

$$\mathbf{X} = \frac{\mathbf{X_{old}} - \overline{\mathbf{X}}}{\sigma_{\mathbf{X}}} \tag{2.24}$$

See Also

`FCMclust, GKclust, GGclust`

PCA

Purpose

Principal Component Analysis. Projection of the n-dimensional data into a lower q-dimensional data (Section 1.4.1).

Syntax

`[result] = PCA(result,data,param)`

Description

The inputs of the PCA function are the multidimensional $data.X$ and the $param.q$ projection dimension parameter (with a common value of 2). The function calculates the autocorrelation matrix of the data and its eigenvectors, sort them according to eigenvalues, and they are normalized. Only the first eigenvector of q-dimension is selected, it will be the direction of the plane to which the data points are projected. The output matrices are saved in $result.PCAproj$ structure, one can get the projected data (P), the projected cluster centers(vp), the eigenvectors (V) and the eigenvalues (D). The results are evaluated with projeval function.

Example

The example was generated with the *PCAexample* function located in *..\Demos\PCAexample\* directory, where the *nDexample* function is situated, which generated the random 3-D data.

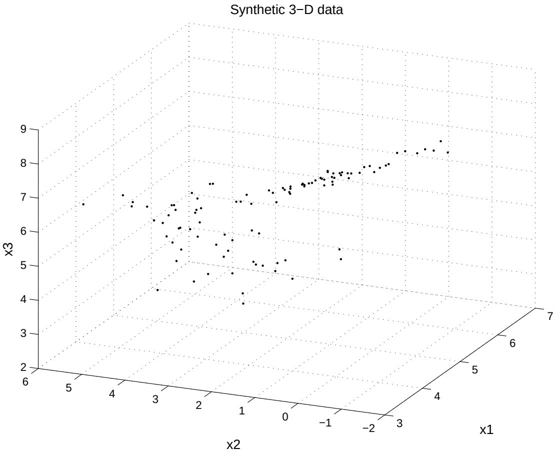

Figure 2.13: Random generated 3-dimensional data set.

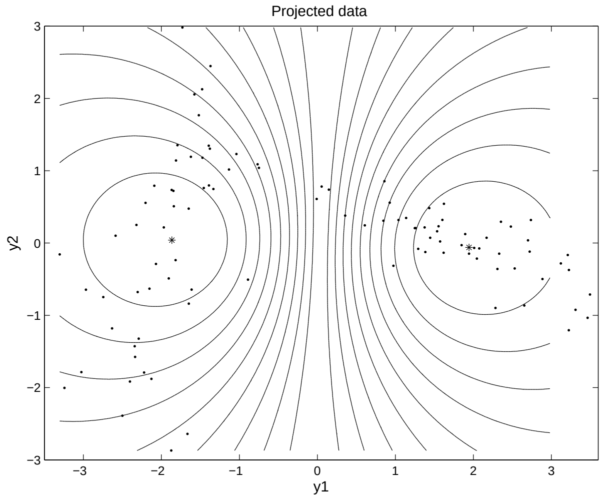

Figure 2.14: The into 2-dimension projected data by Principal Component Analysis.

Discussion

A simple example is presented: random generated 3-D data is clustered with Fuzzy C-means algorithm, and through PCA projection the results are plotted. The original and the projected data is shown in Fig. 2.13 and Fig. 2.14. The error (defined at the description of the projeval function on page 53) is P = 0.0039.

Notes

The PCA MATLAB function is taken from the SOM Toolbox, which is obtainable on the web:
http://www.cis.hut.fi/projects/somtoolbox/.

See Also

`projeval, samstr`

References

Juha Vesanto, Neural Network Tool for Data Mining: SOM Toolbox,*Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000)*, 184-196, 2000.

Sammon

Purpose

Sammon's Mapping. Projection of the n-dimensional data into a lower q-dimensional data (described in Section 1.4.2).

Syntax

[**result**] = `Sammon`(**result**,**data**,**param**)

Description

The Sammon function calculates the original Sammon's Mapping. It uses $result.data.d$ of the clustering as input and two parameters: the maximum iteration number ($param.max$, default value is 500.) and the step size of the gradient method ($param.alpha$, default value is 0.4.). The $proj.P$ can be given either an initializing projected data matrix or the projection dimension. In latter case the function calculates with random initialized projected data matrix, hence it needs normalized clustering results. During calculation the Sammon function uses online drawing, where the projected data points are marked with 'o', and the projected cluster centers with '*'. The online plotting can be disengaged by editing the code, if faster calculation wanted. The results are evaluated with projeval function.

Example

The examples would be like the one shown in Fig. 2.14 on page 49.

Notes

The Sammon MATLAB function is is taken from the SOM Toolbox, which is obtainable on the web:
http://www.cis.hut.fi/projects/somtoolbox/.

See Also

`projeval, samstr`

References

Juha Vesanto, Neural Network Tool for Data Mining: SOM Toolbox,*Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000)*, 184-196, 2000.

FuzSam

Purpose

Fuzzy Sammon Mapping. Projection of the n-dimensional data into a lower q-dimensional data (described Section 1.4.3).

Syntax

```
[result] = FuzSam(proj,result,param)
```

Description

The FuzSam function modifies the Sammon Mapping, so it becomes computationally cheaper. It uses $result.data.f$ and $result.data.d$ of the clustering as input.It has two parameters, the maximum iteration number ($param.max$, default value is 500.) and the step size of the gradient method ($param.alpha$, default value is 0.4.). The $proj.P$ can be given either an initializing projected data matrix or the projection dimension. In latter case the function calculates with random initialized projected data matrix, hence it needs normalized clustering results. During calculation FuzSam uses online drawing, where the projected data points are marked with 'o', and the projected cluster centers with '*'. The online plotting can be disengaged by editing the code. The results are evaluated with projeval function.

Example

The examples would be like the one shown in Fig. 2.14 on page 49.

Algorithm

- $[\textbf{Input}]$: Desired dimension of the projection, usually $q = 2$, the original dataset, $\mathbf{X}$; and the results of fuzzy clustering: cluster prototypes, $\mathbf{v_i}$, membership values, $\mathbf{U} = [\mu_{ki}]$, and the distances $D = [d_{ki} = d(\mathbf{x}_k, \mathbf{v_i})]_{N\times c}$.
- $[\textbf{Initialize}]$ the projected datapoints by $\mathbf{y}_k$ PCA based projection of $\mathbf{x}_k$, and compute the projected cluster centers by

$$\mathbf{z}_i = \frac{\sum_{k=1}^{N}(\mu_{ki})^m \mathbf{y}_k}{\sum_{k=1}^{N}(\mu_{ki})^m} \tag{2.25}$$

  and compute the distances with the use of these projected points $D^* = [d^*_{ki} = d(\mathbf{y}_k, \mathbf{z}_i)]_{N\times c}$

```
• [While] ($E_{fuzz} > \varepsilon$) and (t$\leq$ maxstep)
  {for ($i = 1 : i \leq c : i++$
   {for ($j = 1 : j \leq N : j++$
    {Compute $\frac{\partial E(t)}{\partial y_{il}(t)}$, $\frac{\partial^2 E(t)}{\partial^2 y_{il}(t)}$, $\Delta y_{il} = \Delta y_{il} + \left[ \frac{\frac{\partial E(t)}{\partial y_{il}(t)}}{\frac{\partial^2 E(t)}{\partial^2 y_{il}(t)}} \right]$ }
   }
   $y_{il} = y_{il} + \Delta y_{il} \forall i = 1, \ldots, N, \ l = 1, \ldots, q$
   Compute $\mathbf{z}_i = \sum_{k=1}^{N} (\mu_{ki})^m \mathbf{y}_k / \sum_{k=1}^{N} (\mu_{ki})^m$
   $D^* = [d^*_{ki} = d(\mathbf{y}_k, \mathbf{z}_i)]_{N \times c}$
  }
  Compute $E_{fuzz}$ by (1.46).
```

where the derivatives are defined in (1.45).

See Also

`projeval, samstr`

References

A. Kovács - J. Abonyi, Vizualization of Fuzzy Clustering Results by Modified Sammon Mapping, *Proceedings of the 3rd International Symposium of Hungarian Researchers on Computational Intelligence*, 177-188, 2002.

projeval

Purpose

Evaluation for projected data.

Syntax

[**perf**] = projeval(**result**,**param**)

Description

The projeval function uses the results and the parameters of the clustering and the visualization functions. It is analog to the clusteval function, but it evaluates the projected data. The distances between projected data and projected cluster centers are based on the Euclidean norm, so the function calculates only with a 2-by-2 identity matrix, generates the pair-coordinate points, calculates the new partition matrix, and draws a contour-map by selecting the points with the same partitioning values.
A subfunction of projeval is to calculate relation-indexes defined on the ground of (1.48):

$$P = \overline{\|\mathbf{U} - \mathbf{U}^{*}\|}, \quad \sum_{k=1}^{N} \overline{\mu_k^2}, \quad \sum_{k=1}^{N} \overline{\mu_k^{2*}}. \tag{2.26}$$

See Also

`clusteval, PCA, Sammon, FuzSam, samstr`

samstr

Purpose

Calculation of Sammon's stress for projection methods.

Syntax

```
[result] = samstr(data,result)
```

Description

The simple function calculates Sammon's stress defined in (1.43). It uses $data.X$ and $result.proj.P$ as input, and the only output is the $result.proj.e$ containing the value of this validation constant.

See Also

`PCA,Sammon, FuzSam`

Chapter 3

Case Studies

The aim of this chapter is present the differences, the usefulness and effectiveness of the partitioning clustering algorithms by partitioning different data sets. In Section 3.1 the five presented algorithms are compared based on numerical results (validity measures). Section 3.2 deals with the problem of finding the optimal number of clusters, because this information is rarely known *apriori*. A real partitioning problem is presented in Section 3.3 with three real data sets: different types of wine, iris flowers and breast cancer symptoms are partitioned.

3.1 Comparing the clustering methods

Using the validity measures mentioned in Section 1.3 the partitioning methods can be easily compared. To solve this problem, a synthetic data set was used shown in Fig. 3.1-5, so the index-values are better demarcated at each type of clustering. These validity measures are collected in Tab. 3.1.

First of all it must be mentioned, that all these algorithms use random initialization, so different runnings issue in different partition results, i.e. values of the validation measures. On the other hand the results hardly depend from the structure of the data, and no validity index is perfect by itself for a clustering problem. Several experiment and evaluation are needed that are not the proposition of this work. The presented figures were generated with the *Kmeanscall, Kmedoidcall, FCMcall, GKcall, GGcall* functions, which are located in the ..*Demos**comparing*\ directory. Each function call *modvalidity* function, which calculates all the validity measures, and it exists also in the directory above.

Table 3.1: **The numerical values of validity measures**

	PC	CE	SC	S	XB	DI	ADI
K-means	1	NaN	0.095	0.0001	3987.4	0.0139	0.0004
K-medoid	1	NaN	0.2434	0.0003	Inf	0.0037	0.0036
FCM	0.8282	0.3470	0.9221	0.0008	19.6663	0.0175	0.0119
GK	0.8315	0.3275	0.8697	0.0009	32.1243	0.0081	0.0104
GG	0.9834	0.0285	2.2451	0.0020	2.5983	0.0160	0.0084

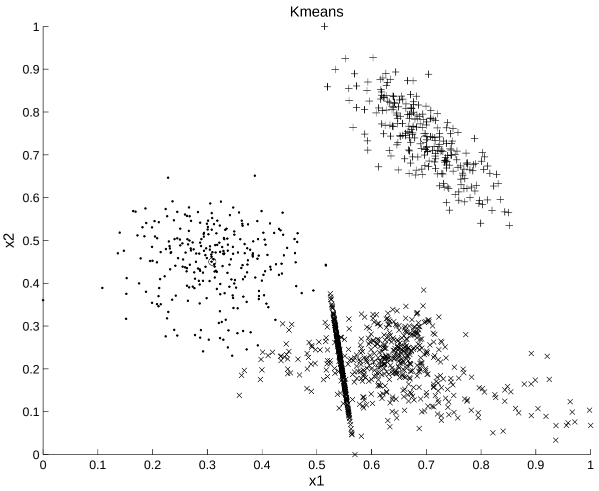

Figure 3.1: Result of K-means algorithm by the synthetic overlapping data with normalization.

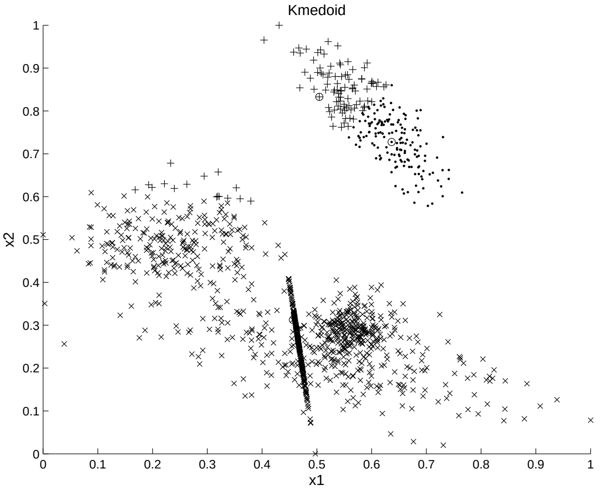

Figure 3.2: Result of K-medoid alogrithm by the synthetic overlapping data with normalization.

Fig. 3.1 shows that hard clustering methods also can find a good solution for the clustering problem, when it is compared with the figures of fuzzy clustering algorithms. On the contrary in Fig. 3.2 one can see a typical example for the initialization problem of hard clustering. This caused the differences between the validity index values in Tab. 3.1, e.g. the Xie and Beni's index is infinity (in "normal case" the K-medoid returns with almost the same results as K-means). The only difference between Fig. 3.3 and Fig. 3.4 stands in the shape of the clusters, while the Gustafson-Kessel algorithm can find the elongated clusters better (the description can be find in Section 1.2.3and the concrete algorithm on page 37). Fig. 3.5 shows that the Gath–Geva algorithm returned with a result of three subspaces.

As one can see in Tab. 3.1, PC and CE are useless for K-means and K-medoid, while they are hard clustering methods. But that is the reason for the best results in S, DI (and ADI), which are useful to validate crisp and well separated clusters.
On the score of the values of the two "most popular and used" indexes for fuzzy clustering (Partition Coefficient and Xie and Beni's Index) the Gath-Geva clustering has the very best results for this data set.

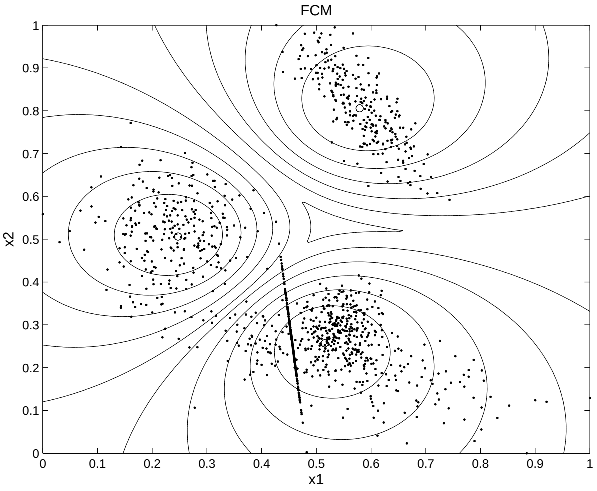

Figure 3.3: Result of Fuzzy C-means algorithm by the synthetic overlapping data with normalization.

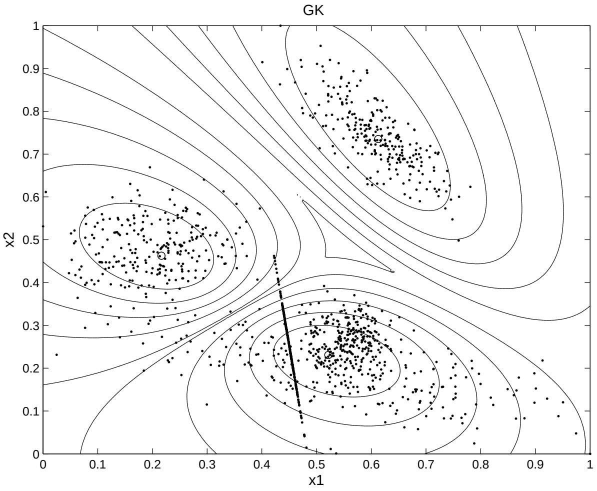

Figure 3.4: Result of Gustafson-Kessel algorithm by the synthetic overlapping data with normalization.

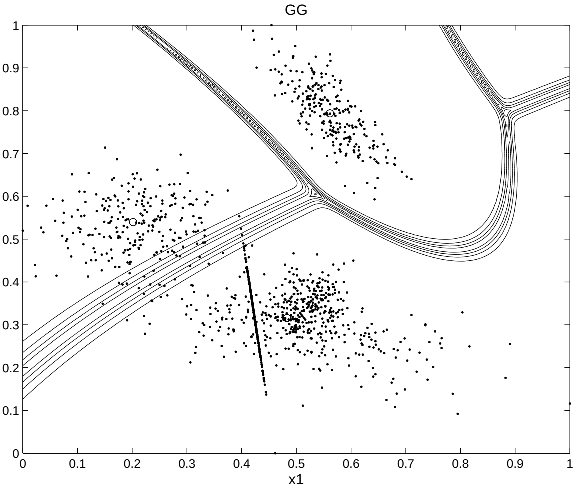

Figure 3.5: Result of Gath-Geva algorithm by the synthetic overlapping data with normalization.

3.2 Optimal number of clusters

In the course of every partitioning problem the number of subsets (called the clusters) must be given by the user before the calculation, but it is rarely known *apriori*, in this case it must be searched also with using validity measures. In this section only a simple example is presented: the motorcycle data set is used to find out the optimal number of clusters. In Section 1.3 described validity measures were used with Gustafson-Kessel algorithm to validate the partitioning of the motorcycle data with the current number of clusters. The results are discussed. The presented figures were generated with the *optnumber* function, which is located in the ..\Demos\optnumber\\ directory, and it calls also the *modvalidity* function like the calling functions in Section 3.1.

During the optimalization parameters were fixed to the following values: $m = 2$, $\varepsilon = 0.001$, $\rho = 1$ for each cluster, $c \in [2\ 14]$. The values of the validity measures depending from the number of cluster are plotted and embraced in Tab. 3.2.

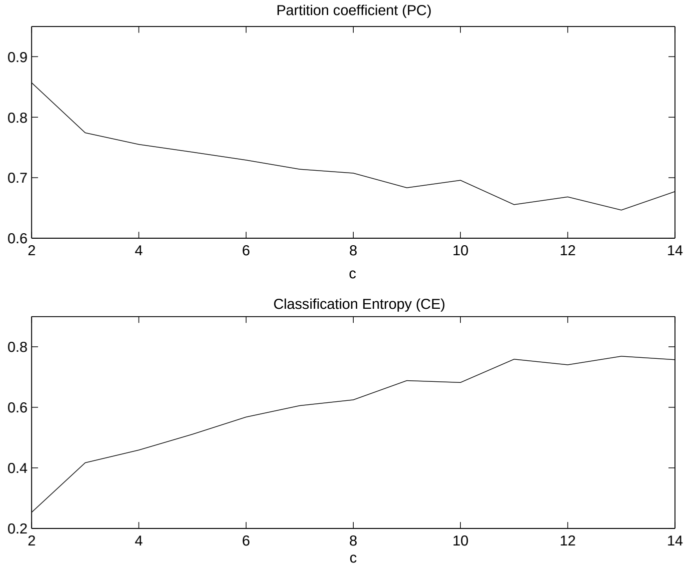

Figure 3.6: Values of Partition Coefficient and Classification Entropy

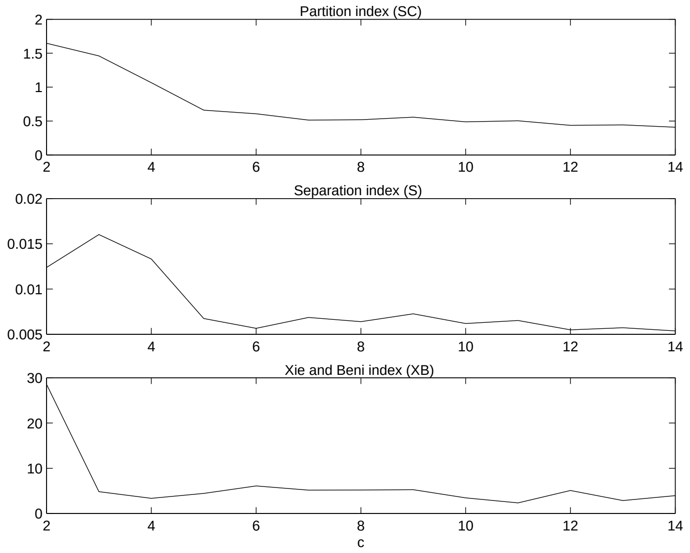

Figure 3.7: Values of Partition Index and Separation Index and Xie and Beni's Index

We must mention again, that no validation index is reliable only by itself, that is why all the programmed indexes are shown, and the optimum can be only detected with the comparison of all the results. We consider that partitions with less clusters are better, when the differences between the values of a validation index are minor.

The main drawback of PC is the monotonic decreasing with c and the lack of direct connection to the data. CE has the same problems: monotonic increasing with c and hardly detectable connection to the data structure. On the score of Fig. 3.6, the number of clusters can be only rated to 3.

In Fig. 3.7 there are more informative diagrams are shown: SC and S hardly decreases at the $c = 5$ point. The XB index reaches this local minimum at $c = 4$. Considering that SC and S are more useful, when comparing different clustering methods with the same c, we chose the optimal number of clusters to 4, which is confirmed by the Dunn's index too in Fig. 3.8. (The Alternative Dunn Index is not tested enough to know how reliable its results are.)

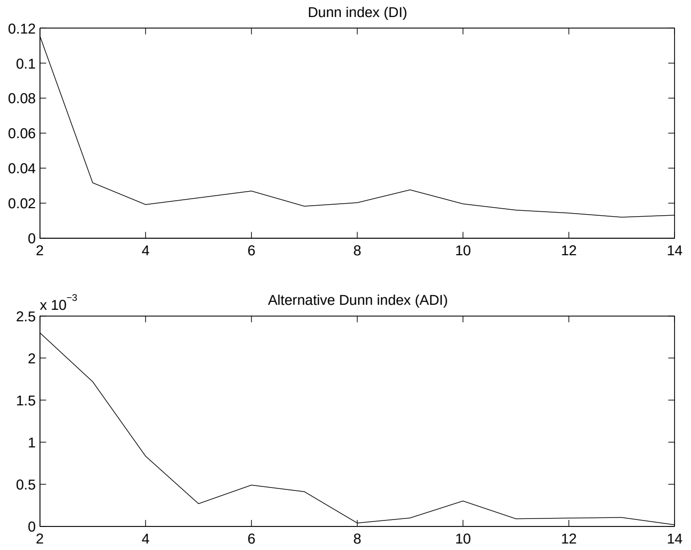

Figure 3.8: Values of Dunn's Index and the Alternative Dunn Index

Table 3.2: **The numerical values of validity measures**

c	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
PC	0.8569	0.7743	0.7550	0.7422	0.7291	0.7139	0.7075	0.6833	0.6957	0.6554	0.6682	0.6464	0.6771
CE	0.2531	0.4168	0.4588	0.5110	0.5679	0.6055	0.6246	0.6882	0.6820	0.7590	0.7404	0.7689	0.7572
SC	1.6465	1.4591	1.0646	0.6593	0.6055	0.5126	0.5181	0.5565	0.4868	0.5032	0.4354	0.4427	0.4079
S	0.0124	0.0160	0.0133	0.0067	0.0057	0.0069	0.0064	0.0072	0.0062	0.0065	0.0055	0.0057	0.0054
XB	28.5271	4.8294	3.3545	4.4216	6.0929	5.1619	5.1688	5.2679	3.4507	2.3316	5.0879	2.8510	3.9492
DI	0.1154	0.0317	0.0192	0.0230	0.0270	0.0183	0.0203	0.0276	0.0196	0.0160	0.0144	0.0120	0.0132
ADI	0.0023	0.0017	0.0008	0.0003	0.0005	0.0004	0.0000	0.0001	0.0003	0.0001	0.0001	0.0001	0.0000

3.3 Multidimensional data sets

In order to examine the performance of the proposed clustering methods three well-known multidimensional classification benchmark problems are presented in this section.

- Iris data
- Wine data
- Wisconsin Breast Cancer data

These data sets come from the UCI Repository of Machine Learning Databases, and they are downloadable from:
ftp://ftp.ics.uci.edu/pub/machine-learning-databases/, but they also exist in the *..\Demos\projection\* directory. Cause of the too many data points there is no use to show the partition matrixes in tables, so the results of the n-dimensional clustering was projected into 2-dimension, and the 2-D results were plotted. Considering that projected figures are only approximations of the real partitioning results, the difference between the original and the projected partition matrix is also represented, and on the other hand one can observe the difference between the PCA, Sammon's mapping and the Modified Sammon Mapping too, when these values are comprehended. These relation-indexes are defined in (2.26).

Because of using real data sets the classes are known, so the misclassed objects can be plotted. These items are signed with an 'o' accessory-marker. The goodness of the classification is also can be defined by the percental error of these misclassed data points (a data point is misclassed when on the grounds of its largest membership degree it is not in the cluster which it belongs to):

$$Err = \frac{N_{missel}}{N} \cdot 100. \tag{3.1}$$

Note that calculating Err indicates only the goodness of the classification not the clustering, hence we must take its results under protest. This error value is independent from the visualization method, so they are stated below in Tab. 3.3. For the comparison five independent run were estimated with each algorithm. Fuzzy C-means, Gustafson-Kessel and Gath-Geva returned always with the same minimum, while the results of K-means and K-medoid hardly depend from the initialization as shown in Tab. 3.3, so in the case of these algorithms the minimum, mean and maximum were also represented.
The difference between the minimal and maximal value shows the sensitivity of the algorithm to the initialization (as one can see K-medoid is the most sensitive

	Kmeans			Kmedoid			FCM	GK	GG
-	min	mean	max	min	mean	max	-	-	-
Iris	10.667	10.933	11.333	9.333	25.199	34.667	10.667	10	20
Wine	5.056	5.505	6.742	7.865	24.494	39.887	5.056	38.202	24.719
Wisc	3.806	3.923	3.9531	2.635	3.982	4.8316	3.367	9.809	11.420

Table 3.3: Percental values of the misclassed objects with different clustering methods.

algorithm). The minimal value denotes the flexibility of the model, the mean denotes the estimation of the expected error. One can see that the best stable results has the Fuzzy C-means clustering for these data sets, so its resulting figures are shown in the following subsections, which are generated with the *visualcall* function (situated in the same directory as the data sets).

3.3.1 Iris data set

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other. Predicted attribute: class of iris plant.

The attributes are as follows:x_1 - sepal length in cm, x_2 - sepal width in cm, x_3 - petal length in cm, x_4 - petal width in cm. In x_5 attribute are the three classes: Iris Setosa (marked with '.'), Iris Versicolour (marked with 'x') and Iris Virginica(marked with '+'). The projected cluster centers are marked with '*'.

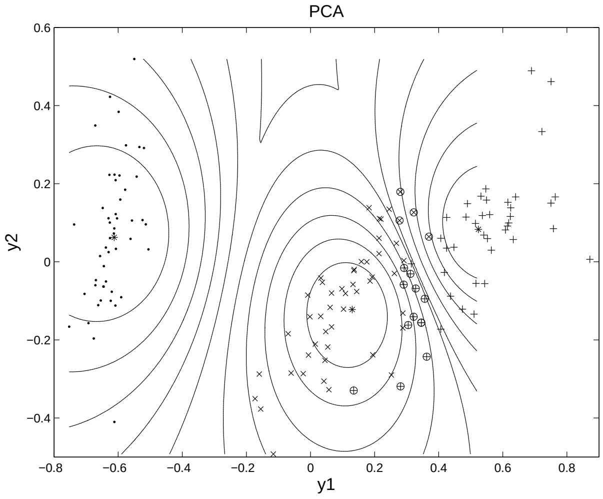

Figure 3.9: Result of PCA projection by the Iris data set.

	P	$\sum_{k=1}^{N} \overline{\mu_k^2}$	$\sum_{k=1}^{N} \overline{\mu_k^{2*}}$	E
PCA	0.0203	0.7420	0.7850	0.0117
Sammon	0.0132	0.7420	0.7662	0.0071
FuzSam	0.0025	0.7420	0.7426	0.0131

Table 3.4: Relation-indexes on Iris data set.

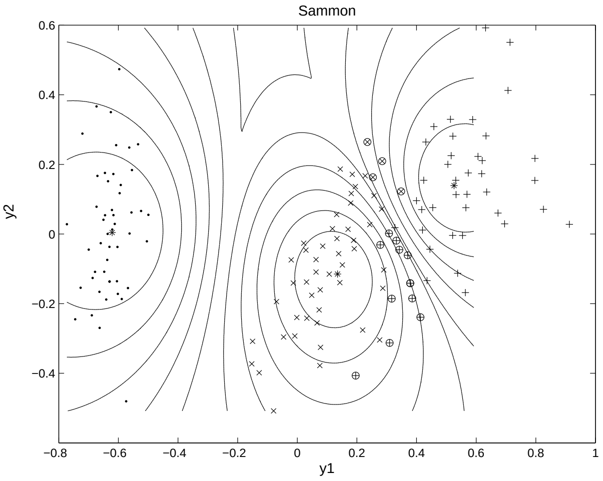

Figure 3.10: Result of Sammon's mapping projection by the Iris data set.

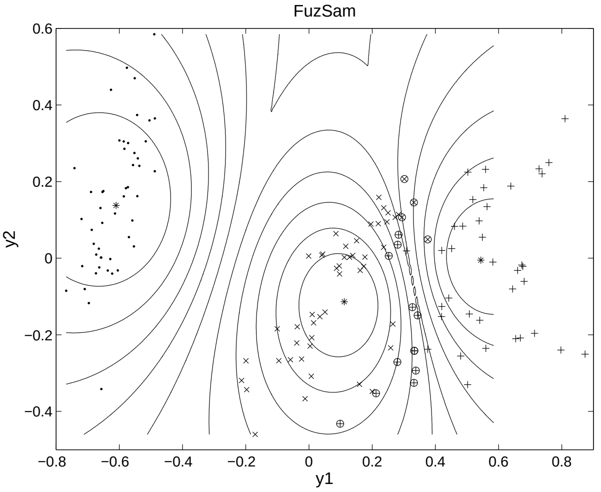

Figure 3.11: Result of Fuzzy Sammon Mapping projection by the Iris data set.

3.3.2 Wine data set

The Wine data contains the chemical analysis of 178 wines grown in the same region in Italy but derived from three different cultivars (marked with '.','x' and '+'). The problem is to distinguish the three different types based on 13 continuous attributes derived from chemical analysis.

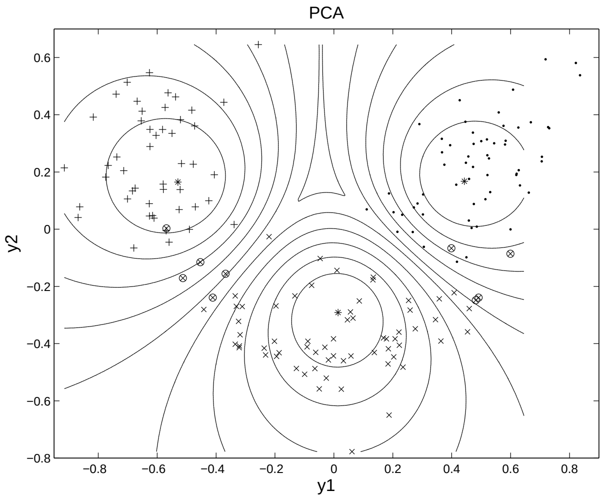

Figure 3.12: Result of PCA projection by the Wine data set.

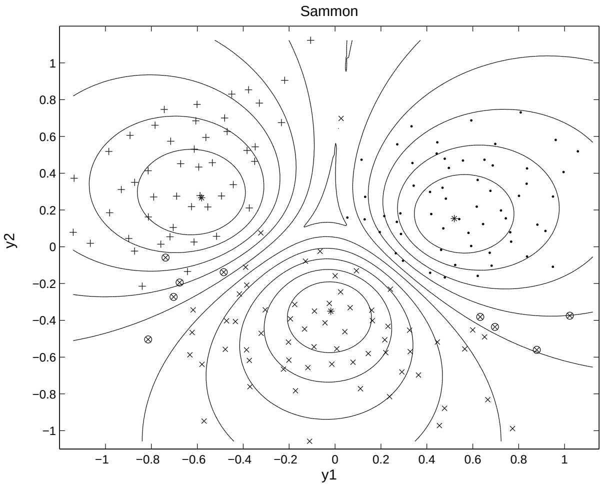

Figure 3.13: Result of Sammon's mapping projection by the Wine data set.

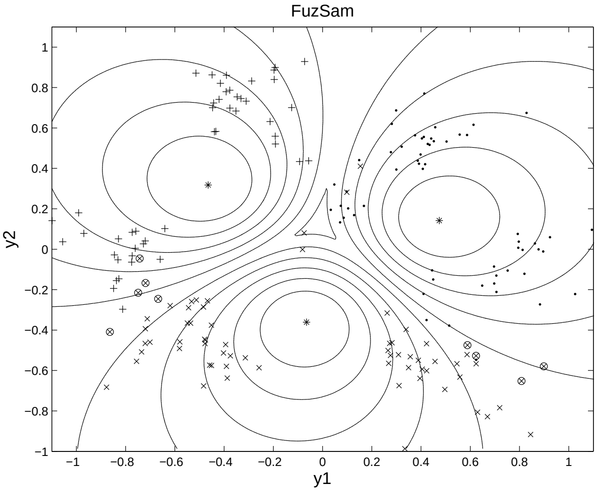

Figure 3.14: Result of Fuzzy Sammon Mapping projection by the Wine data set.

	$\mathbf{P}$	$\sum_{k=1}^{N} \overline{\mu_k^2}$	$\sum_{k=1}^{N} \overline{\mu_k^{2*}}$	E
PCA	0.1295	0.5033	0.7424	0.1301
Sammon	0.0874	0.5033	0.6574	0.0576
FuzSam	0.0365	0.5033	0.5170	0.0991

Table 3.5: Relation-indexes on Wine data set.

3.3.3 Breast Cancer data set

The Wisconsin breast cancer data is widely used to test the effectiveness of classification and rule extraction algorithms. The aim of the classification is to distinguish between *benign* and *malignant* cancers based on the available nine measurements: x_1 clump thickness, x_2 uniformity of cell size, x_3 uniformity of cell shape, x_4 marginal adhesion, x_5 single epithelial cell size, x_6 bare nuclei, x_7 bland chromatin, x_8 normal nuclei, and x_9 mitosis. The attributes have integer value in the range [1,10]. The original database contains 699 instances however 16 of these are omitted because these are incomplete, which is common with other studies. The class distribution is 65.5% benign and 34.5% malignant, respectively.

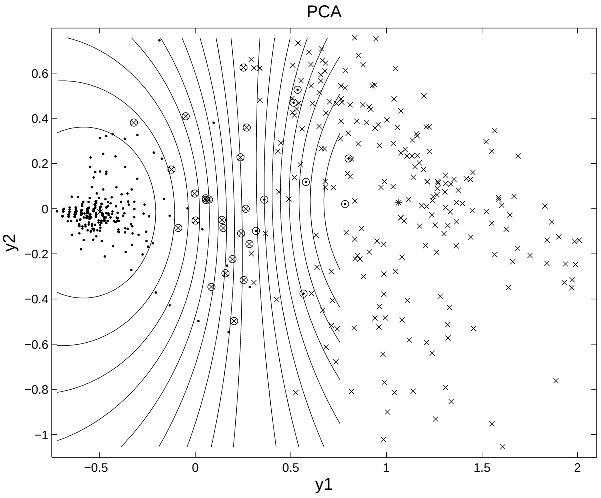

Figure 3.15: Result of PCA projection by the Wisconsin Breast Cancer data set.

	P	$\sum_{k=1}^{N}\overline{\mu_k^2}$	$\sum_{k=1}^{N}\overline{\mu_k^{2*}}$	E
PCA	0.0456	0.8409	0.9096	0.0882
Sammon	0.0233	0.8409	0.8690	0.0260
FuzSam	0.0050	0.8409	0.8438	0.0497

Table 3.6: Relation-indexes on Wisconsin Breast cancer data set.

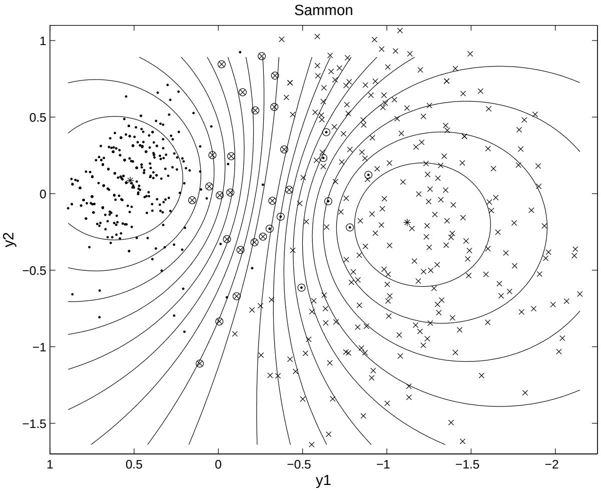

Figure 3.16: Result of Sammon's Mapping projection by the Wisconsin Breast Cancer data set.

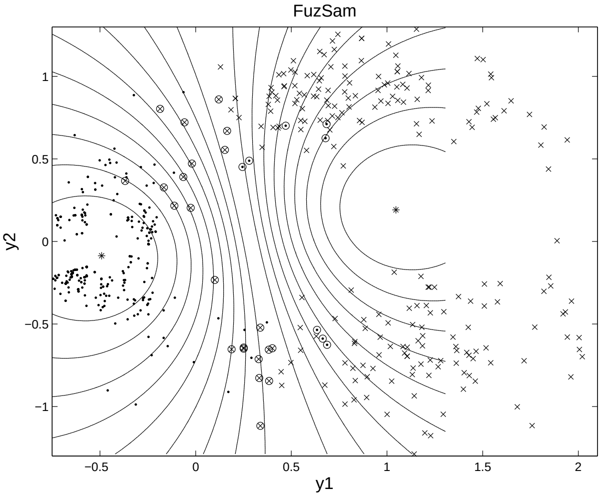

Figure 3.17: Result of Fuzzy Sammon Mapping projection by the Wisconsin Breast Cancer data set.

3.3.4 Conclusions

As Tab. 3.4, Tab. 3.5 and Tab. 3.6 show, Fuzzy Sammon Mapping has much better projection results by the value of $\mathbf{P}$ than Principal component Analysis, and it is computationally cheaper than the original Sammon Mapping. So during the evaluation of the partition the figures created with this projection method were considered. We calculated the original Sammon's stress for all the three techniques to be able to compare them.

Tab. 3.3 shows that the "advanced" algorithms does not have always the best results. It depends on the underlying structure of the data. The Gustafson-Kessel and the Gath-Geva algorithm cannot detect correctly the wine species, as it is shown in Fig. 3.18. On the contrary the Fuzzy C-means provides dependable results for all data sets. By far the best results are in the case of the Wisconsin Breast Cancer data, where from 683, 9-dimensional points "only" 23 are misclustered, and the 96.6 % secure result is very reliable, considering that the clustering techniques do not use prior class identifiers, the are among the unsupervised methods.

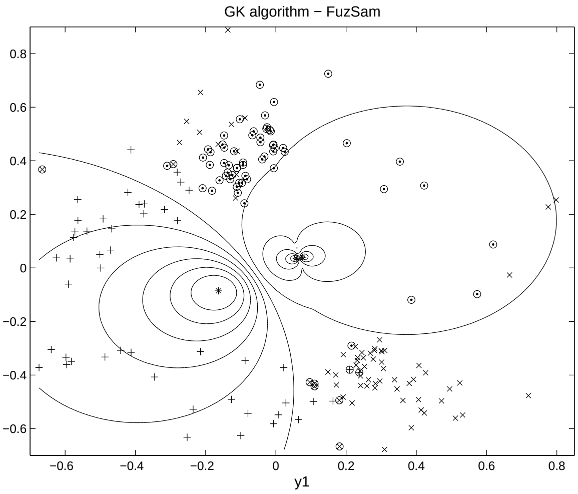

Figure 3.18: Result of Fuzzy Sammon projection by the wine data set with Gustafson-Kessel algorithm.

Summary

To meet the growing demands of systematizing the nascent data, a flexible, powerful tool is needed. The Fuzzy Clustering and Data Analysis Toolbox provides several approaches to cluster, classify and evaluate wether industrial or experimental data sets. The software for these operations has been developed with MATLAB, which is very powerful for matrix-based calculations. The Toolbox provides five different types of clustering algorithms, which can be validated by seven validity measures. High-dimensional data sets can be also visualized with a 2-dimension projection, hence the toolbox contains three different method for visualization.
With all these useful tools the work is not finished yet: many more clustering algorithms, validation indices, visualization methods exist or will come up the next day. They could take place in this work too, hence the author knows *"it is not the end... it is the beginning..."*

Bibliography

[1] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.

[2] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. Wiley, Chichester, 1999.

[3] R. Babuška. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Boston, 1998.

[4] D.E. Gustafson and W.C. Kessel. Fuzzy clustering with fuzzy covariance matrix. In *Proceedings of the IEEE CDC, San Diego*, pages 761–766. 1979.

[5] R. Babuška, P. J. van der Veen, and U. Kaymak. Improved covariance estimation for Gustafson-Kessel clustering. *IEEE International Conference on Fuzzy Systems*, pages 1081–1085, 2002.

[6] J.C. Bezdek and J.C. Dunn. Optimal fuzzy partitions: A heuristic for estimating the parameters in a mixture of normal dustrubutions. *IEEE Transactions on Computers*, pages 835–838, 1975.

[7] I. Gath and A.B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:773–781, 1989.

[8] A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, and R.F. Murtagh. Validity-guided (Re)Clustering with applications to imige segmentation. *IEEE Transactions on Fuzzy Systems*, 4:112–123, 1996.

[9] X. L. Xie and G. A. Beni. Validity measure for fuzzy clustering. *IEEE Trans. PAMI*, 3(8):841–846, 1991.

[10] Sammon JW Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.

[11] M. Aladjem I. Dinstein B. Lerner, H. Guterman and Y. Romem. On pattern classification with sammon's nonlinear mapping - an experimental study. *Pattern Recognition Letters*, 31(4):371–381, 1998.

[12] Juha Vesanto. Neural network tool for data mining: Som toolbox. In *Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000)*, pages 184–196. 2000.

[13] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal components analysis. *Neural Computation*, 11:443–482, 1999.

[14] Kovács A.-Abonyi J. Vizualization of fuzzy clustering results by modified sammon mapping. In *Proceedings of the 3rd International Symposium of Hungarian Researchers on Computational Intelligence*, pages 177–188. 2002.