Visual Data Analysis in Air Traffic Management

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik

der Otto-von-Guericke-Universität Magdeburg

von FRANK REHM, M.Sc.

geboren am 25. März 1976 in Dresden

Gutachter:

Prof. Dr. Rudolf Kruse

Prof. Dr. Frank Klawonn

Prof. Dr. Karl Nachtigall

Magdeburg, den 12. Februar 2007

Zusammenfassung

In nahezu allen Bereichen von Handel, Dienstleistung, Industrie und Forschung werden große Mengen von Daten gesammelt. Der Grund dafür liegt meist in dem Wunsch, das Verhalten von Kunden zu verstehen sowie technische oder natürliche Phänomene beschreiben bzw. vorhersagen zu können.

Neben der Größe solcher Datensätze, die eine Analyse ohne Unterstützung von Computern unmöglich macht, stellt die Anzahl der Attribute, die ein Datenobjekt beschreiben, eine Herausforderung dar. Während Datenobjekte, die durch zwei oder drei Attribute beschrieben sind, einfach graphisch dargestellt werden können, ist die Visualisierung hoch-dimensionaler Daten – also Daten, die durch sehr viele Attribute beschrieben werden – nicht trivial.

Das Forschungsgebiet Data Mining umfasst die Entwicklung geeigneter Methoden zur Datenaufbereitung und Datenanalyse vor dem Hindergrund wachsender Datenbanken mit komplexen Datensätzen. Diese Arbeit liefert einen Beitrag auf dem Gebiet der Methodenentwicklung zur Dimensionsreduktion und Ausreißererkennung. Ein wesentlicher Beitrag besteht in der Visualisierung komplexer Daten, sowie der Visualisierung von Ergebnissen verbreiteter statistischer Analysemethoden, wie Clustering oder Fuzzy-Klassifikatoren.

Am Beispiel der Analyse von Flug- und Wetterdaten vom Flughafen Frankfurt wird deutlich, welche Stärken und welche Grenzen die in dieser Arbeit vorgestellten Methoden charakterisieren. In diesem Beispiel soll der Einfluss des Wetters auf die Flugzeit ankommender Flugzeuge am Frankfurter Flughafen bestimmt werden. Dadurch soll die Vorhersage von Flugzeiten möglich werden, was die Optimierung verschiedener Abläufe am Flughafen zulässt.

Abstract

Almost all branches of commerce, industry and research put great efforts in collecting data with the objective to describe and predict customer behaviour or both technical and natural phenomena.

Besides the size of such data sets, which make manual analysis impractical, data analysis becomes challenging due to a large number of attributes describing a data object. Whereas a graphical representation of data objects that are described by means of two or three attributes can be realized easily, the visualization of high-dimensional data – data that is described through many attributes – is not trivial.

The data mining research area comprises the development of suitable techniques for data preprocessing and data analysis to cope with the problem of aggrandizing databases including complex data sets. This thesis contributes to the domain of methodology development, dimensionality reduction and outlier treatment. Another major focus is set on the visualization of complex data as well as the visualization of complex results obtained from common data mining techniques, e.g. clustering and fuzzy classifiers.

The characteristics of the proposed techniques become evident on the example of the analysis of flight data and weather data measured at Frankfurt Airport. The objective of this application is the research of weather factors that affect the flight duration of aircraft approaching Frankfurt Airport. Understanding the interrelationship between weather and flight duration permits the optimization of various processes at the respective airport and may save time and money of customers and companies.

Contents

1 Introduction

Nowadays, data collecting is practiced in almost every domain of business and science. Data mining or knowledge discovery is the hidden agenda behind this process. Ideally, one would input all the collected data into a black box which outputs all the knowledge that comprehends the data. Unfortunately, this is not available until now. Indeed, the process of knowledge extraction is all but trivial. First problems arise when experiments or measurements produce data of low quality. Data mining algorithms must thus be able to deal with uncertainty or imprecision.

Most classical data mining methods expect a homogeneous input. In many modern applications, however, the data to be analyzed come from heterogeneous information sources. We certainly cannot expect to find data mining algorithms that are generally applicable to all kinds of information sources.

Due to his excellent capability of visual pattern recognition, a human can easily group data into clusters or classify different phenomena simply by viewing them on a sheet of paper or on a computer screen. This hypothesizes admittedly that the nature of the problem representation is 3-dimensional at most. However, the problems we focus in this work are naturally high-dimensional in a virtual feature space and thusly not analyzable directly by viewing only.

On the one hand, the issue of this work will be the visualization of problems that can be represented in a high-dimensional feature space. But we will also provide some improvements to common techniques that mine information from suchlike data.

1.1 Data Mining and Visualization

The analysis of collected data is not a new activity. Statisticians have been defining mathematical descriptions of data for many years. Research work in statistical analysis, pattern recognition and machine learning all contribute to data mining. Often the available data comprise only a sample from the complete population. The aim may be to generalize from the sample to the population. Such generalizations may not be achievable through standard statistical approaches because often the data are not random samples, but rather represent a biased subsample [39].

According to [28] data mining is the mechanized process of identifying or discovering useful structures in data. The term structure refers to patterns, models or relations over the data. A pattern is described as a description of a subset of data points. A model is a description of the entire data set. A relation is an accurate, convenient and useful summary of some aspect of the data specifying some dependency between attributes over a subset of the data. Typically favoured requirements of relations we seek are, of course, novelty, but also simplicity. Relationships must be understandable to be accepted by a user.

Two general categories in data mining are prediction and knowledge discovery. Prediction involves using some variables to predict unknown or future values of other variables of interest. Prediction will be commonly distinguished between classification, which implies the prediction of discrete variables, and regression that works on continuous variables [16]. Knowledge discovery focuses on finding interpretable patterns describing the data. It has been defined as the nontrivial extraction of implicit, previously unknown, interesting and potentially useful information from data.

Visual methods are important in data mining because they help to provide comprehension of unexpected relationships. The goal is to reduce complexity while capturing important information. Visualizations can be used to explore data, to confirm or deny a hypothesis or to emphasize certain aspects of the data to the viewer. In exploratory visualizations the user is searching

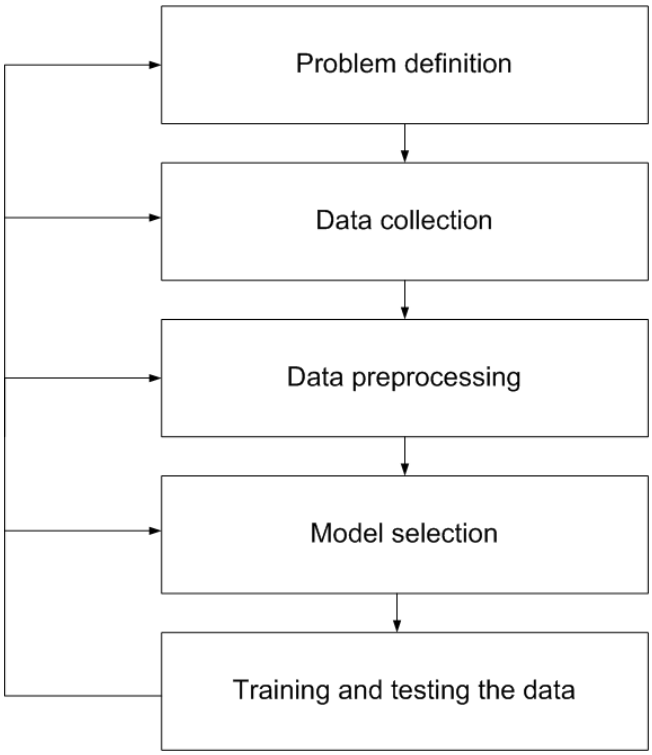


Figure 1.1: The data mining process

for structure or trends attempting to formulate some hypotheses. Visualization can be static, animated or interactive. Without the proper visualization techniques, data mining models may not give us the desired insight to help humans understand the phenomena of interest [28]. A detailed survey of visualizations is given in [42].

The data mining process consists of different stages that are visualized in figure 1.1 and summarized in the following. *The problem definition* stage includes the description of the form of input and output, but also costs and the appropriateness of using data mining. *Data collection* is concerned with deciding which data to collect and how to collect them. *The data preprocessing* task assures that data conforms to a certain format and duplicates or outliers are treated appropriately. *Selecting an appropriate mining method* consists of selecting an algorithm, e.g. regression, clustering, etc., and of setting algorithm

parameters, such as the number of prototypes for the clustering. *Training and testing the data* means the training of the mining algorithm and testing it using an evaluation set of data in the trained architecture. A detailed examination of the data mining process can be explored in [16, 39, 90].

This work will mainly contribute to the data preprocessing stage, the model selection stage and the training and testing stage. For that purpose we will describe some techniques for outlier treatment and visualization techniques that help to select suitable mining methods. Further we contribute to the visual evaluation of clustering algorithms and to the visual presentation of fuzzy rules.

1.2 Improving Air Traffic Management with Data Mining

Air traffic growth has been stated in the past decades neglecting a temporary decrease in 2001–2002. Likewise traffic forecast estimates an increase of about 4% yearly for Europe [73]. The big effort that is made to lower the delay of aircraft emphasizes that often airports already operate at their limit [1, 2, 26]. A serious problem is that heavy aircraft induce wake vortices[1] that constrain separation to succeeding aircraft. Inventing procedures to reduce separation is subject of current research. Studies have shown that delay is also caused to a big extent by bad weather [13, 26, 40, 71, 93]. Thunderstorms must be dodged, heavy headwind reduces flight speed and iced aircraft must be de-iced – just to mention some reasons for that. Besides the weather forecast that is usually released by meteorological services also the prediction of the delay under given weather conditions might improve the airport efficiency. Of course, the impact of weather on air traffic cannot be generalized for all airports since local particularities vary naturally. Predicting the delay that an aircraft may have allows to retard flights on other airports that depart to the airport but also to coordinate ground activities such as baggage handling.

[1] Wake vortices are small tornadoes, counter-rotating, being generated at the end of the wings of an aircraft.

Analyzing a weather data set with the objective to predict aircraft delay will be a thread in this thesis.

In [40] arriving flights at Los Angeles International Airport (LAX) are focused. A metric – the Daily Flight Time Index (DFTI) – is developed that captures the daily variation in flight times for LAX arrivals. The day-to-day variation in DFTI is analyzed relating it to weather, demand, and average delays at origin airports. A linear regression model is applied to data which describes several weather factors, traffic information and an origin airport delay variable. The model explains 75% of the variation in DFTI. Several other models are estimated and compared to this baseline model.

The impact of thunderstorms at Frankfurt Airport has been studied in [93]. Thunderstorm days were compared with adequate reference days with normal weather conditions. The difference in the delay times between a thunderstorm day and the reference day was determined on an hourly basis. A significant increase in delay was observed. Since the total delay depends on the intensity and duration of the thunderstorm event and the instant capacity of the airport, delay is varying strongly. In a period of two years delay factors ranged between 0.1 and 12.7.

1.3 Data Presentation

Studies in this thesis are based either on some synthetic data sets, on well known benchmark data sets and on two combined data sets coming from an industrial application. The last contains on the one hand the weather conditions at Frankfurt Airport and the complete traffic data for the same time period on the other hand. Details of the synthetic data and the benchmark data will be given later. A brief description of the weather data and the traffic data will be given in the following. For detailed information on both data sets we refer to [61, 78].

Weather Data

The weather data originate from the ATIS[2] weather data set of the year 1998: different sensors present at the airport capture several weather characteristics and form a weather report. Such a report is released every thirty minutes (in case of rapidly changing weather, the frequency is increased); this corresponds to over 18000 data. Each weather report contains information such as the temperature, the air pressure and precipitation information, e.g. the presence of snow, rain or hail.

Traffic Data

In addition to the weather data set, information about the traffic is available, through a data set that contains the arrival times of all aircraft at Frankfurt Airport for the observed time period. Since the delay is of interest that is caused by the weather factors in the vicinity of the airport, the point in time of the aircraft's entrance in the airport vicinity (the so-called Terminal Area, TMA) and the time when the corresponding aircraft is landing are considered. The difference between these two times corresponds to the travel time in the TMA. This quantity is independent of delay that might have been caused at the departure airport or during the flight. In average, the TMA travel time is about thirty minutes.

Whereas the studies in [61, 76, 77, 78] aimed at analyzing the effect of certain weather factors to flight durations, we will demonstrate the efficiency of our techniques by means of these data sets.

1.4 Data Preprocessing

Advances in data collection and storage capabilities during the past decades have led to an information overload in most sciences. In most of the cases when it is to solve a task one gets raw data that have to be prepared for

[2] ATIS (Automatic Terminal Information Service) is a continuous broadcast of recorded information in airports. ATIS broadcasts contain essential weather information but also the active runway and other information required by the pilots.

concrete learning processes. Before a data set will be formatted to conform to a certain analysis tool it is essential to determine appropriate data structures and then the data preprocessing method. Some important aspects in this regard are:

- data cleaning
- data transformation
- data reduction.

Alike the weather data set requires all these preprocessing steps.

Data cleaning covers the handling of missing values, the identification and processing of outliers and resolving inconsistencies. These steps are important since outliers as well as inconsistencies have a bad impact on a multitude of learning methods. In the recent years, much work has been done to cope with the outlier problem [3, 4, 12, 38, 52, 53]. Data cleaning is far from trivial since suchlike (missing) data can be the result of erroneous measurement but it can also comprehend important information [4, 96]. Missing data can be filled in using imputation-based or model-based techniques. Replacing missing values without capturing the information that they were missing actually removes information from the data set. The most predictive variable in a data set could be the missing value pattern. For an extensive discussion of missing value treatment we refer to [97]. Thus, the traffic data set needs to be cleaned because of some outlying values due to very heavy traffic and due to accidents in two cases during the observed time period. The weather data set contains missing values when certain weather conditions are fulfilled but also in case of the malfunction of different sensors.

Data transformations are needed for different reasons, amongst others for aggregation and normalization. Often it may be sufficient to classify data by means of linguistic terms, e.g. travel time is *low*, *medium* or *high*. In most of the cases different attributes vary in range. Normalization is then advisable in order not to affect learning methods in an unjustified way. As a matter of fact, not all learning methods process every type of data. Transformations, e.g. dichotomization or binarization of nominal attributes will be necessary

to process data. Actually, the weather data set contains a nominal attribute which describes precipitation by one or more facts, e.g. *RASH* which means *rain shower* whereas *RA* simply indicates *rain*. Constructing new features in such cases inflates the data set's complexity drastically.

Data reduction and dimensionality reduction may be essential to be able to analyze data within an acceptable timeframe under maintainable resources. Moreover, not all the measured variables are important for understanding the underlying phenomena of interest. Thus, the problem is to identify relevant features. The goal is to reduce complexity while losing the least amount of information. Commonly used methods like principal component analysis or multidimensional scaling provide dimensionality reduction. Multiple regression, regression trees or feature selection techniques facilitate the identification of indispensable variables. Sampling, which selects a representative subset from a large population of data, is needed when analyzing large data sets that overextend resources, be it time or memory storage. As a preprocessing step for data mining, discretization consists of splitting the values of a continuous variable into a small list of intervals. Each interval is then treated as a discrete value by the data mining algorithm. An effect of discretization is to speed up the execution of several algorithms, e.g. rule induction algorithms [16].

1.5 Outline

In this work we advance the methodology regarding data visualization and data mining of high-dimensional data. Regarding the data mining process, we will mainly contribute to the stages of data preprocessing, mining algorithms and model evaluation. In chapter 2, we review multidimensional scaling and focus on three own techniques. We apply our methods to synthetic data and on real weather data as well to visualize them on the plane. Chapter 3 deals with the improvement of fuzzy clustering and the visualization of fuzzy partitions. Concerning clustering we aim at making existing techniques more robust. Validity of fuzzy partitions is difficult to determine.

Our visualizations target to identify single clusters. In chapter 4, we review the concept of fuzzy rules and describe a way to visualize high-dimensional data in relation to their representing rule base. Finally, we conclude with chapter 5.

2 Multidimensional Scaling and Data Navigation

Multidimensional scaling (MDS) is a family methods that seek to present the important structure of the data in a reduced number of dimensions[3]. For this purpose MDS estimates the coordinates of a set of objects $Y = \{y_1, \ldots, y_n\}$ in a feature space of specified (low) dimensionality that come from data $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ trying to preserve the distances between pairs of objects. Different ways of computing distances and various functions relating the distances to the actual data are commonly used. These distances are usually stored in a distance matrix

$$D^x = \left(d_{ij}^x\right), \quad d_{ij}^x = \left\|x_i - x_j\right\|, \quad i, j = 1, \ldots, n.$$

The estimation of the coordinates will be carried out under the constraint, that the error (or *stress*) between the distance matrix D^x of the data set and the distance matrix $D^y = \left(d_{ij}^y\right)$, $d_{ij}^y = \left\|y_i - y_j\right\|$, $i, j = 1, \ldots, n$ of the corresponding transformed data set will be minimized.

2.1 Sammon's Mapping

Different error measures to be minimized were proposed, e.g. the absolute error, the relative error or a combination of both. A commonly used error measure, the so-called *Sammon's mapping*

$$E = \frac{1}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}^x} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{\left(d_{ij}^y - d_{ij}^x\right)^2}{d_{ij}^x} \tag{2.1}$$

[3]Principal component analysis can be regarded as a basic form of multidimensional scaling.

Algorithm 1 Sammon's mapping

Given the data set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$
Define projection dimension $q \in \{1, \ldots, p-1\}$
Define step size α and the threshold value E^*
Compute $d_{ij}^x, \quad i, j = 1, \ldots, n$
Initialize $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^q$
repeat
  Compute $d_{ij}^y, \quad i, j = 1, \ldots, n$
  Compute $\partial E / \partial y_k, \quad k = 1, \ldots, n$
  $y_k = y_k - \alpha \cdot \frac{\partial E}{\partial y_k}, \quad k = 1, \ldots, n$
until $\sum_{k=1}^{n} \left(\partial E / \partial y_k\right)^2 < E^*$
Output projected data set $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^q$

describes the absolute and the relative quadratic error [91]. To determine the transformed data set Y by means of minimizing error E a gradient descent method can be used. By means of this iterative method, the parameters y_k to be optimized, will be updated during each step proportional to the gradient of the error function E. Calculating the gradient of the error function leads to

$$\frac{\partial E}{\partial y_k} = \frac{2}{\sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}^x} \sum_{j \neq k} \frac{d_{kj}^y - d_{kj}^x}{d_{kj}^x} \frac{y_k - y_j}{d_{kj}^y}. \tag{2.2}$$

After random initialization for each projected feature vector y_k a gradient descent is carried out and the distances d_{ij}^y as well as the gradients $\frac{\partial d_{ij}^y}{\partial y_k}$ will be recalculated again. Algorithm 1 that shows the procedure with pseudo code terminates when E becomes smaller than a certain threshold E^*.

The complexity of MDS is $O(c \cdot n^2)$, where c is the (unknown) number of iterations needed for convergence of the gradient descent scheme. Thus, MDS is usually not applicable to larger data sets. Another problem of MDS is that it does not construct an explicit mapping from the high-dimensional space to the lower dimensional space, but just tries to position the lower dimensional feature vectors in a suitable way. Therefore, when new data have to be con-

sidered, they cannot be mapped directly to the lower dimensional space, but the whole MDS procedure has to be repeated.

2.2 Modern Multidimensional Scaling

In the past decade much research work has been done to improve multidimensional scaling. Most effort has been accomplished to reduce the time complexity of the algorithm in order to permit the application of MDS to larger data sets. There are too many approaches to review them all here, but we will give a brief overview on some important representatives.

Chalmers proposed in [14] a stochastically-based algorithm of linear complexity per iteration to produce low-dimensional layouts. Instead of doing all the possible pairwise gradient calculations as in Sammon's mapping, this method carries out gradient calculations between feature vector x_i and the members of two sets whose size is bounded by a constant. One set contains a dynamically maintained list of references to neighbour objects. Entries in this set are stored in order of distance in the high-dimensional space. This neighbour set is carried over between iterations. The second set, which is a randomly chosen subset of all objects, is constructed anew each iteration and has no member of the neighbour set. Candidate elements for the second set will be inserted to the neighbour set instead if the distance to feature vector x_i is lower than the distance of one or more of the current neighbours. In this way, the neighbour set becomes more representative of the most similar objects to x_i over successive iterations. Stress calculation is performed every $\sqrt{n}$ iterations. With this method Chalmers reports the applicability to larger data sets. The overall costs of this approach are $O(n^2)$ because the total number of iterations depends on the data set size as given in [14].

FastMap [27] approaches multidimensional scaling through the projection of objects on a carefully selected arbitrary low-dimensional hyper-plane. The key idea is to assume that the data are in some unknown high-dimensional space. Only a distance matrix is given. A heuristic strategy is used to determine a line between two *pivot objects* which allows projections with little

information loss. Thus, the task is to find a line on which the projections are as far apart from each other as possible. To achieve this, pivot objects are chosen such that the distance between them is maximized. The proposed heuristic strategy is linear in the number of objects alike the projection on lower dimensions since distance preservation is only arranged to pivot objects. FastMap achieves significant time savings over MDS at the expense of higher stress for a given target dimensionality. To obtain low stress FastMap has to map the data set to a fairly higher feature space than conventional MDS.

In [67] an improvement of Chalmers' algorithm is proposed which has $O(n\sqrt{n})$ time complexity. In this work a $\sqrt{n}$ sample of the data set is taken to build Chalmers' model. The complete layout of the entire data set is performed by means of an interpolation strategy. The placement of an object begins with finding the most similar member of the initial layout and then finding the best position on the circumference of the circle whose radius is defined by the original distance between the object to place and the most similar member. This position is then refined by iteratively adding gradients from a subset of the data, moving the object until its final location. Further refinements have been proposed in [66].

The MDSteer algorithm, proposed in [99], iteratively alternates between a layout stage in which a sub-selection of points are added to the set of active points affected by the MDS iteration, and a binning stage which increases the depth of the bin hierarchy and organizes the currently unplaced points into separate spatial regions. This binning strategy allows to select regions of the layout to focus the MDS computation into the areas of the data set that are assigned to the selected bins. The authors emphasize the steerability to MDS. The complexity of MDSteer is comparable to Chalmers' approach. The main benefit is the ability of interactive investigation of data sets with high-dimensionality.

In [69, 70] an approach is proposed that reduces computational complexity by means of using Sammon's mapping to map only some representatives of the data set which are obtained using a clustering algorithm. The remaining

data points are mapped using a relative MDS procedure that considers only distances between the cluster centres and the data points leading to a reduced number of computations.

Many other approaches related to multidimensional scaling have been published in the recent years for which we refer to the literature [10, 23, 24, 43, 95, 103].

2.3 MDS_{polar}

MDS_{polar}, a new approach that we have first described in [80], is about the 2-dimensional projection of a p-dimensional data set X. MDS_{polar} tries to find a representation in polar coordinates $Y = \{(l_1, \varphi_1), \ldots, (l_n, \varphi_n)\}$, where the length l_k of the original vector x_k is preserved and only the angle φ_k has to be optimized. Thus, our solution is defined to be optimal if all angles between pairs of data objects in the projected data set Y coincide as good as possible with the angles between data objects X in the original feature space.

A straight forward definition of an objective function to be minimized for this problem would be

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} (|\varphi_i - \varphi_k| - \psi_{ik})^2 \tag{2.3}$$

where φ_k is the angle of y_k, ψ_{ik} is the positive angle between x_i and x_k, $0 \leq \psi_{ik} \leq 180^\circ$. E is minimal, if the differences of the angles of all pairs of vectors of data set X and the corresponding two vectors in data set Y are zero. The absolute value is chosen in equation (2.3) because the order of the minuends can have an influence on the sign of the resulting angle. The problem with this notation is that the functional E is not differentiable, exactly in those points we are interested in, namely, where the term $|\varphi_i - \varphi_k|$ becomes zero. Another meaningful approach would be

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} ((\varphi_i - \varphi_k)^2 - \psi_{ik}^2)^2. \tag{2.4}$$

In this case the derivative can be easily determined, however, resulting in a

system of nonlinear equations for which no analytical solution can be provided.

In order to overcome these difficulties, we propose an efficient method that enables us to compute an approximate solution for a minimum of the objective function (2.3) and related ones. In a first step we ignore the absolute value in (2.3) and consider

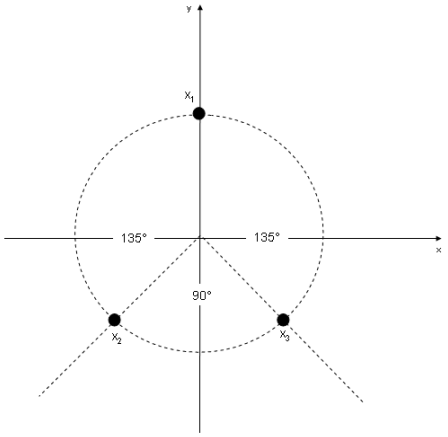
$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - \psi_{ik})^2 \tag{2.5}$$

instead. When we simply minimize (2.5), the results will not be acceptable. Although the angle between y_i and y_k might perfectly match the angle ψ_{ik}, $\varphi_i - \varphi_k$ can either be ψ_{ik} or $-\psi_{ik}$. Since we assume that $0 \leq \psi_{ik}$ holds, we always have $(|\varphi_i - \varphi_k| - \psi_{ik})^2 \leq (\varphi_i - \varphi_k - \psi_{ik})^2$. Therefore, finding a minimum of (2.5) means that this is an upper bound for the minimum of (2.3). Therefore, when we minimize (2.5) in order to actually minimize (2.3), we can take the freedom to choose whether we want the term $\varphi_i - \varphi_k$ or the term $\varphi_k - \varphi_i$ to appear in (2.5). Before we discuss techniques to minimize (2.5) with the freedom of reordering, we have to preprocess the data in order to fit them best to our approach.

2.3.1 Data Preprocessing

Figure 2.1 illustrates an important problem by means of a simple data set. The table next to the graphics contains the values of the angles between the three feature vectors.

Even though, this feature space has only two dimensions and therefore an exact reproduction of the data set should be possible, this cannot be achieved without additional preprocessing. Since we only want to preserve the angles between data vectors, it is obvious that any solution will be invariant with respect to rotation of the data set. Thus, assuming without loss of generality $\varphi_1 = 0$ enforcing $\varphi_2 = 135$, then according to our objective function (2.3) $\varphi_3 = 180$ leads to the optimal solution, which is obviously not what we are looking for. This problem is caused by the fact that ψ_{ik} is defined as a



ψ	x_1	x_2	x_3
x_1	0	135	135
x_2	135	0	90
x_3	135	90	0

Figure 2.1: A preprocessing step for MDS_{polar}

positive angle which satisfies $\psi_{ik} \leq 180°$. This problem can be solved easily by translating all feature vectors into the first quadrant. More generally, for a high-dimensional data set we apply a translation that makes all components of data vectors non-negative. For this we only have to determine for each component the largest negative value occurring in the data set and using this as a positive value of the corresponding component of the translation vector. Note that, when the data set is normalized, i.e. all components are between 0 and 1, no further preprocessing is required.

Thus, doing this kind of preprocessing, we actually do not preserve the original data properties but those after the transformation. Of course, rotation and translation is not changing any inter-data properties. The translation vector has to be stored so that for incremental adding of new objects the transformation can be performed accordingly. For most of the new objects the transformation will be as requested. It may occur that for new objects which have one or more extreme components the translation will not be sufficient to eliminate the negative components. In such a case, which is rather rare if the previous data is representative, the mapping of the respective object is still working, but might lead to non-optimal solutions.

2.3.2 Approximation of MDS_{polar}

When we are free to choose between $\varphi_i - \varphi_k$ and $\varphi_k - \varphi_i$ in (2.5), we take the following into account

$$(\varphi_k - \varphi_i - \psi_{ik})^2 = (-(\varphi_k - \varphi_i - \psi_{ik}))^2 = (\varphi_i - \varphi_k + \psi_{ik})^2.$$

Therefore, instead of exchanging the order of φ_i and φ_k, we can choose the sign of ψ_{ik}, leading to

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - a_{ik}\psi_{ik})^2 \tag{2.6}$$

with $a_{ik} = \{-1, 1\}$. In order to solve this modified optimization problem of equation (2.6) we take the partial derivatives of E, yielding

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} (\varphi_i - \varphi_k - a_{ik}\psi_{ik}). \tag{2.7}$$

Thus, on the one hand, neglecting that we still have to choose a_{ik}, our solution is described by a system of linear equations which means its solution can be calculated directly without the need of any iteration procedure. On the other hand, as described above, we have to handle the problem of determining the sign of ψ_{ik} in the form of the a_{ik}-values.

To fulfil the necessary condition for a minimum we set equation (2.7) equal to zero and solve for the φ_k-values, which leads to

$$\varphi_k = \frac{\sum_{i=1}^{k-1} \varphi_i - \sum_{i=1}^{k-1} a_{ik}\psi_{ik}}{k-1}. \tag{2.8}$$

Different optimization strategies are conceivable. Of course, an important condition is the computational complexity of the respective approximation algorithm. In this work we describe a number of different strategies, starting with a greedy algorithm which is quadratic with the number of data objects in time, but is linear in space. Later on, we propose an algorithm that can even reduce the complexity to $O(n \cdot \log n)$.

2.3.3 A Greedy Algorithm for the Approximation of MDS_{polar}

As mentioned above, the solution of MDS_{polar} is described by a system of linear equations. Since the minimization problem (see equation 2.6) is rotation invariant, i.e. any rotation of a solution will also minimize (2.6), φ_1 can be set to any value, e.g. $\varphi_1 = 0$. By means of a greedy algorithm we choose $a_{ik} \in \{-1, 1\}$ such that for the resulting φ_k the error E of the objective function (2.6) is minimal. For φ_2 the exact solution can always be found, since a_{12} is the only parameter to optimize. For the remaining φ_k the greedy algorithm sets a_{ik} in turn either -1 or 1, verifying the validity of the result, setting a_{ik} to the better value immediately and continuing with the next a_{ik} until all $k-1$ values for a_{ik} are set.

Algorithm 2 describes in a simplified way the greedy method. When implementing the method, it can be optimized in that way, that the first φ_k in the for-loop has not always to be recalculated if in step $i-1$ the parameter a_{ik} has not been changed to -1. In such cases φ_k keeps the value from the previous step.

As mentioned above, φ_1 can be set to any value and φ_2 can always be chosen in such a way that the angle ψ_{12} is preserved exactly. For the remaining angles φ_k no guarantee can be given that the greedy algorithm finds the optimal solution. Incremental adding of feature vectors can be achieved by simply extending the outer for-loop by another iteration for each new object. The angle φ_k will be computed analogously as for previous feature vectors.

For more accurate transformations (accepting higher calculating times) the inner for-loop can be encapsulated be another loop that enables the algorithm to find better sign configurations that are ignored in the previous loop due to the greedy heuristic. A suchlike loop should be limited by an appropriate constant and can be cancelled immediately if no sign has changed during the last loop. By means of this extension the algorithm is still greedy and yields significantly improved local minimum solutions.

Algorithm 2 Greedy MDS_{polar}

Given the data set $X = \{x_1, x_2, \ldots, x_n\}$
Let $\Psi_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
$\varphi_1 = 0$
for $k = 2$ to n **do**
  $a_{ik} = 1$ for all $i = 1 \ldots k-1$
  for $i = 1$ to $k-1$ **do**
    $\varphi_k = \frac{\sum_{j=1}^{k-1} \varphi_j - \sum_{j=1}^{k-1} a_{jk}\psi_{jk}}{k-1}$ $e_k = \sum_{j=1}^{k-1}(\varphi_j - \varphi_k - a_{jk}\psi_{jk})^2$
    $t = \varphi_k$
    $a_{ik} = -1$
    $\varphi_k = \frac{\sum_{j=1}^{k-1} \varphi_j - \sum_{j=1}^{k-1} a_{jk}\psi_{jk}}{k-1}$ $f_k = \sum_{j=1}^{k-1}(\varphi_j - \varphi_k - a_{jk}\psi_{jk})^2$
    if $e_k < f_k$ **then**
      $a_{ik} = 1$
      $\varphi_k = t$
    end if
  end for
end for

2.3.4 Relative MDS_{polar}

As for conventional MDS, also for MDS_{polar} different approaches regarding the objective function are feasible. The solution described above minimizes the absolute differences of pairwise angles of the original data set and the transformed data set. Large angles, which cause in tendency a large E may affect the solution in that way, that the transformation will represent vectors with small angles to others less correctly. Considering the relative error leads to

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} \left(\frac{\varphi_i - \varphi_k - a_{ik}\psi_{ik}}{\psi_{ik}} \right)^2 \tag{2.9}$$

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} \left(\frac{\varphi_i - \varphi_k - a_{ik}\psi_{ik}}{\psi_{ik}} \right) \frac{1}{\psi_{ik}}. \tag{2.10}$$

The greedy algorithm 2 can be applied only modifying the calculation specification for φ_k

$$\varphi_k = \frac{\sum_{i=1}^{k-1} \frac{\varphi_i}{\psi_{ik}^2} - \sum_{i=1}^{k-1} a_{ik} \frac{1}{\psi_{ik}}}{\sum_{i=1}^{k-1} \frac{1}{\psi_{ik}^2}}. \tag{2.11}$$

Because of the different objective functions the validity of solutions with the absolute MDS_{polar} and the relative MDS_{polar} cannot be compared by means of E.

2.3.5 Weighted MDS_{polar}

In certain cases the objective when transforming data is to preserve relations of feature vectors of the original feature space in the target feature space. Thus, feature vectors that form a cluster should be represented as exact as possible in the target feature space, too. The transformation of feature vectors with a large distance to the respective feature vector can have a lower accuracy. An approach to achieve this goal is the introduction of weights w_{ik} to our objective function

$$E = \sum_{k=2}^{n} \sum_{i=1}^{k-1} w_{ik} (\varphi_i - \varphi_k - a_{ik}\psi_{ik})^2. \tag{2.12}$$

Determining the partial derivative leads to

$$\frac{\partial E}{\partial \varphi_k} = -2 \sum_{i=1}^{k-1} w_{ik} (\varphi_i - \varphi_k - \psi_{ik}) \tag{2.13}$$

and solving for φ_k to

$$\varphi_k = \frac{\sum_{i=1}^{k-1} w_{ik} (\varphi_i - a_{ik}\psi_{ik})}{\sum_{i=1}^{k-1} w_{ik}}. \tag{2.14}$$

Note that this is a generalization of relative MDS_{polar}. For relative MDS_{polar} we simply choose the weights as $w_{ik} = 1/\psi_{ik}^2$.

Since our transformation preserves the length of each data vector, it is guaranteed that vectors with a large difference in length will not be mapped

to close points in the plane, even though their angle might not be matched at all. Therefore, we propose to use a small or even zero weight for pairs of data vectors that differ significantly in their length. The weight could be defined as a function of the difference between the length values l_i and l_j of two data vectors:

$$w_{ik} = w(l_i, l_k) = w(z). \tag{2.15}$$

We can use the absolute difference for z, i.e.

$$z = z_a = |l_i - l_k| .$$

This might be useful if certain information about the structure of the data is known in advance. The argument z_r for relative weighting functions

$$z = z_r = \min \left\{ \frac{l_i}{l_k}, \frac{l_k}{l_i} \right\}$$

might be useful if a certain threshold value can be specified, whose excess excludes the calculation of the angle φ_k between the respective pair of vectors. To decrease the computational complexity, weights should be chosen in such a way, that for feature vectors with a certain (large) distance the respecting weights become zero. The following function describes a simple weighting function, which is the function shown in Figure 2.2(b):

$$w(z_r) = \begin{cases} \sqrt{\left(\frac{z_r - \vartheta}{1 - \vartheta}\right)} & \text{, if} \quad z_r \geq \vartheta \\ 0 & \text{, otherwise} \end{cases} \tag{2.16}$$
$$\text{where} \quad \vartheta \in [0, 1].$$

With the threshold ϑ one can control indirectly the fraction of the data that will be used to determine the respective angle φ_k. Thus, small values for ϑ lead to many weights $w \neq 0$ which is associated with high computational complexity. Values near 1 for ϑ lead to a quickly decreasing weighting function and to low computational complexity, respectively. Any other function can be used as weighting function. For reasons of an easy implementation and low computational complexity a decreasing function which leads to a more or less large fraction of zero weights should be used.

For an efficient implementation it is useful to sort the feature vectors by means of their length. Note that this can be achieved with $O(n \cdot \log n)$ time complexity. When determining the weights for the calculation of φ_k it is sufficient to consider the feature vectors starting from index k. Weights will be calculated stepwise. With every step the weights become smaller until a weight becomes zero. Since the weighting function is decreasing, a further iteration would lead to zero, too. Thus, the calculation of weights stops at this point. In cases where clusters with a large amount of data are expected in a data set, it might rather be useful to limit the maximum number of iterations for the calculation of the weights than setting a larger threshold. In this case, the projected vectors will be forced to a proper position already by a significantly large fraction of other feature vectors in the data set. It might also be useful to reduce ϑ locally, when only few vectors satisfy the condition in equation (2.16).

With a limitation of the number of weights $w > 0$ and a moderate ϑ at the same time, it can be achieved that the number of weights considered for the calculation of φ_k does not differ too much for different φ_k and limited computation time can be guaranteed. Instead of considering the angles of all feature vectors with the greedy algorithm 2 it might be useful to consider only few feature vectors and calculate the exact solution of the sign problem. Using a weighting function enables the user of MDS_{polar} to set a certain bin size which indicates the number of feature vectors that will be considered when calculating the desired φ_k. By means of this one can reduce the computation time and reinvest it in finding the exact solution of the sign problem. Solving the sign problem with the greedy strategy for a given maximum bin size b and using a certain number c of iterations accounts with $O(n \cdot b \cdot c)$ to the algorithm complexity. Thus, the upper bound for the complexity of our algorithm is $O(n \cdot \log n)$ due to sorting the data.

Similar to the idea of stress for standard multidimensional scaling, an evaluation of the transformation can be done by determining the average deviation from the original angles. In general this can be obtained by dividing E by the number of terms summed up. For the error function (2.6) one has to di-

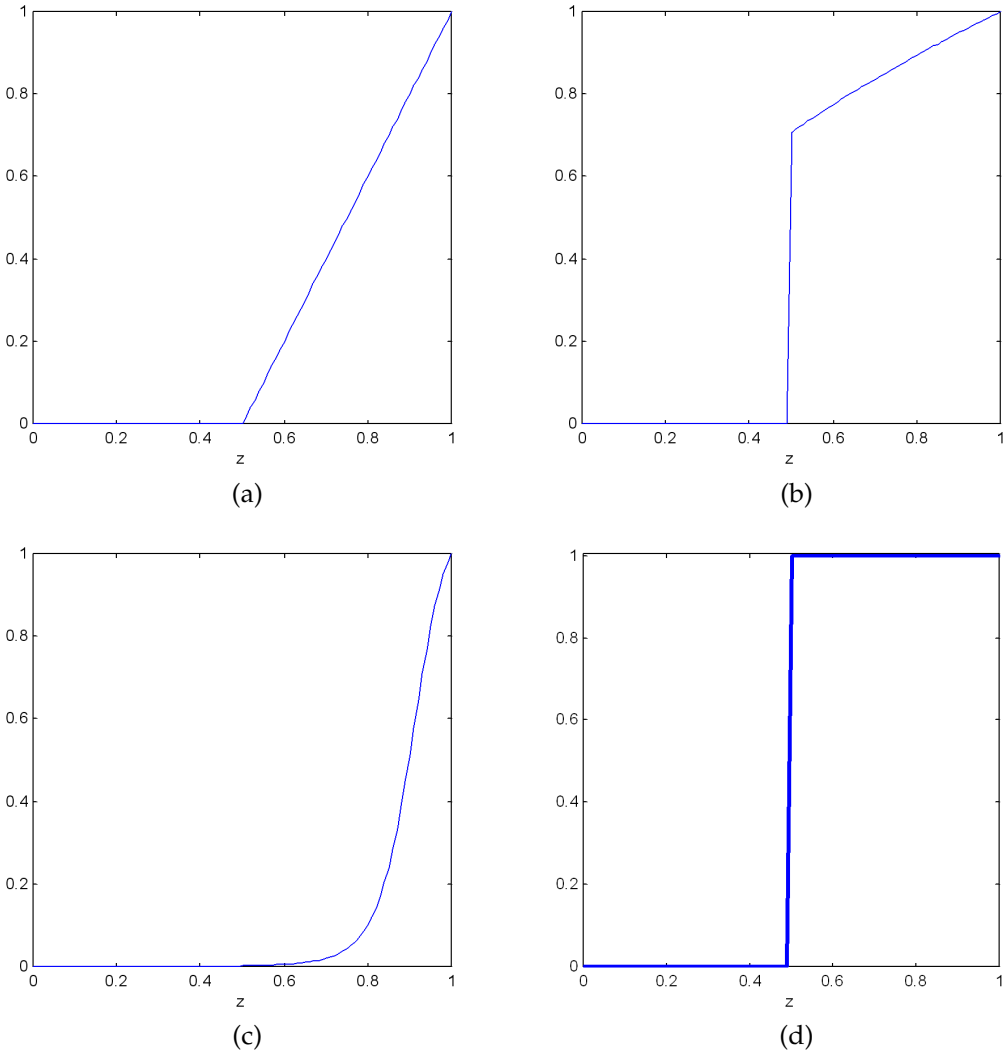
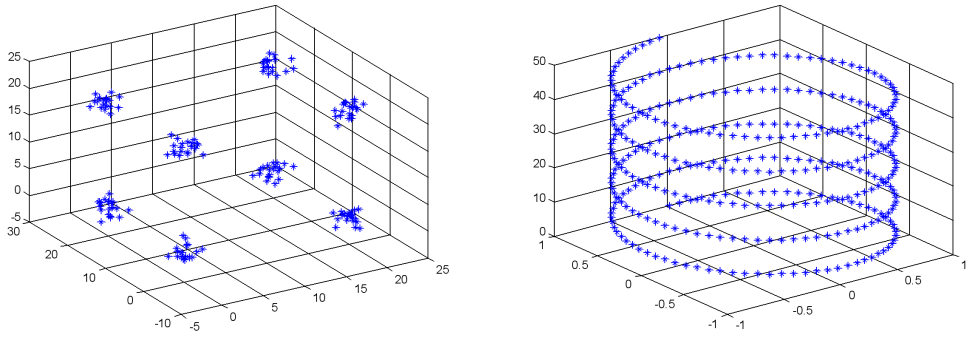


Figure 2.2: Different weighting functions for MDS_{polar}

vide by $\frac{n^2+n}{2}$. With this measured value one can compare different mappings even if they vary in the number of objects.

2.3.6 Illustrative Examples

Figure 2.4 shows some results of MDS_{polar} in comparison with Sammon's mapping. In favour of an easy verification of the results we applied MDS_{polar} to some 3-dimensional data sets. The validity of the solution can be evalu-

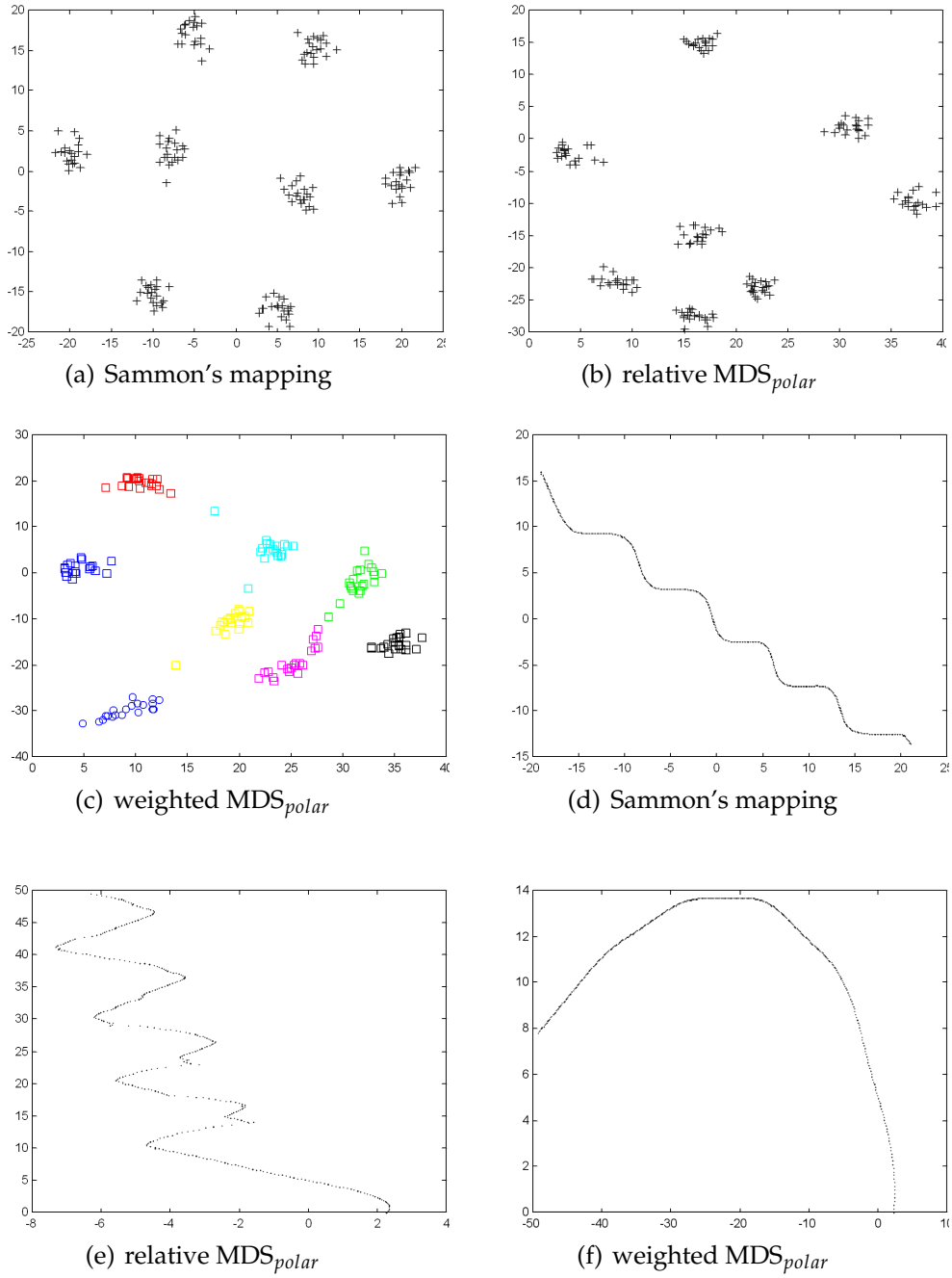


(a) Cube data set (b) Coil data set

Figure 2.3: Two synthetic data sets

ated by visual inspection. The Cube data set (see figure 2.3(a)) is about a synthetic data set, where data points scatter around the corners of an imaginary 3-dimensional cube. Thus, the Cube data set contains eight well separated clusters. The Coil data set (depicted in figure 2.3(b)) is comparable to a serpentine. As figures 2.4(b) and 2.4(c) show, the transformations of MDS_{polar} are similar to these of conventional MDS (figure 2.4(a)). Whereas MDS needs some thousand iterations until convergence, MDS_{polar} finds an explicit solution after solving the system of equations and some hundreds sign adjustments. Figure 2.4(d) shows the Sammon's mapping of the Coil data set. The transformations in figure 2.4(e) and 2.4(f) result from weighted MDS_{polar} with weighting functions where at most twelve weights got values greater than zero. Thus, the transformation is based only on a relatively small number of angle comparisons. Therefore, locally these transformations are very accurate, but generally the loss of information is sometimes higher.

Since the value of φ_k is calculated from all preceding $\varphi_1 \ldots \varphi_{k-1}$ according to equation (2.8) or equation (2.11) respectively, a solution with MDS_{polar}, either absolute or relative, depends to some degree on the order of the data set. Our tests have shown that in such cases only few feature vectors lead to higher errors, while others will not. Thus, not the complete transformation will be wrong, but only some feature vectors. Initialization is also a crucial step for conventional MDS.



(a) Sammon's mapping (b) relative MDS_{polar}

(c) weighted MDS_{polar} (d) Sammon's mapping

(e) relative MDS_{polar} (f) weighted MDS_{polar}

Figure 2.4: Different transformations of synthetic data with MDS_{polar}

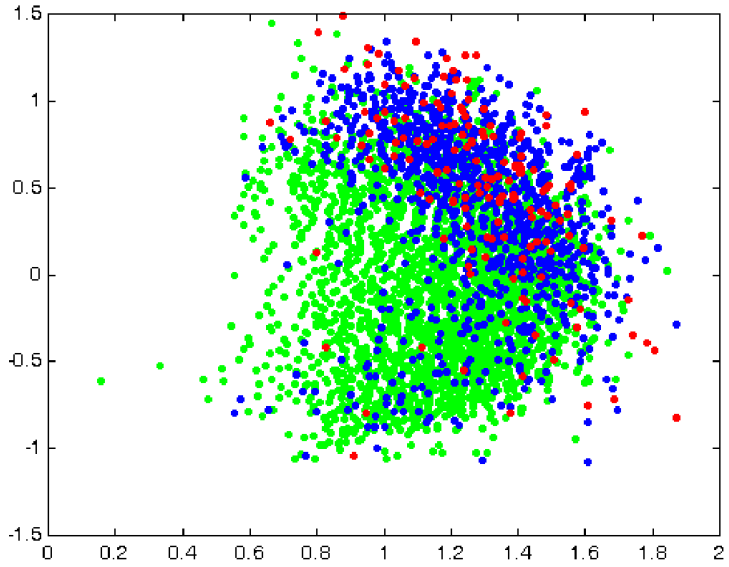


Figure 2.5: Mapping of the weather data with MDS_{polar}

2.3.7 Visualizing Weather Data with MDS_{polar}

Figure 2.5 shows a mapping of non-cloudy weather using relative MDS_{polar} with a bin size of 200. In order to examine the relationships between the weather factors and the flight duration, three classes of travel times are defined: one class represents weather data associated with short travel times with up to one minute more than the average travel time. It is represented in green. A second class, represented in blue, corresponds to medium travel times with up to eight minutes delay compared to the average travel time value. The last class, in red, stands for weather data associated with even later flights. Early flights prevail, less than 10% of the flights are more than eight minutes later than the average. The figure shows clearly that the three classes overlap to some degree. Whereas the green class is spread all over the feature space, the blue class and the red class take the upper right region of the feature space mainly. Figures A.1–A.5 in appendix A show some mappings of the weather data set using various bin sizes.

2.4 POLARMAP

In this section we discuss another approach for dimension reduction that we have presented in [82]. This approach - POLARMAP - is a modification of MDS_{polar}. As for MDS_{polar}, this algorithm transforms high-dimensional feature vectors into 2-dimensional feature vectors under the constraint that the length of each vector is preserved and the angles between vectors approximate the corresponding angles in the original space as good as possible. As an improvement of MDS_{polar}, we will describe an algorithm that learns a function that enables the user to map even new feature vectors to the target space. Finally, we will describe a technique to learn such mappings from data with $O(n \cdot \log n)$ time complexity.

As an extension of MDS_{polar}, POLARMAP is a method that learns a function f that provides for any p-dimensional feature vector x_k the corresponding angle φ_k that is needed to map the feature vector to a 2-dimensional feature space. As for MDS_{polar} the length of vector x_k is preserved. With the obtained function also angles for new feature vectors can be computed. A 2-dimensional scatter plot might not be suitable, when visualizing mappings for large data sets. With the computed function it is simple to produce information murals, which allow more comprehensive visualizations [46].

Analogous to functional (2.3) we define our objective function E as follows:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\left| f(x_i) - f(x_j) \right| - \psi_{ij} \right)^2 . \tag{2.17}$$

E is minimal, if, for each pair of feature vectors, the difference of the two angles, which are computed by the respective function f is equal to the measured angle ψ_{ij} of the two vectors in the original space. Since functional (2.17) is not differentiable, we propose analogous to the procedure for MDS_{polar} to minimize the following differentiable objective function

$$\tilde{E} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(f(x_i) - f(x_j) - \psi_{ij} \right)^2 . \tag{2.18}$$

Since we have always $\psi_{ij} \geq 0$ according to our definition, it is obvious that $E_{min} \leq \tilde{E}_{min}$. Thus, a minimum of $\tilde{E}$ might not be the minimum of E, but it can be used as a conservative estimation. Albeit, f might be any function, we discuss in this work only the following function style

$$f(x) = a^T \cdot \tilde{x} \tag{2.19}$$

where a is vector whose components are the parameters to be optimized and $\tilde{x}$ is feature vector x itself or a modification of x. In the simplest case we use

$$\begin{aligned} \tilde{x} &= x \\ a &= (a_1, a_2, \ldots, a_p)^T \end{aligned} \tag{2.20}$$

where f describes in fact the linear combination of the components of x. Assuming that a certain component of x affects the transformation not linearly but quadratically or exponentially, it may be useful to compute some additional components from x with the objective to gain more coefficients, which could improve the transformation. An example for quadratic components derived from x is described by the following choice:

$$\begin{aligned} \tilde{x} &= (x_1, \ldots, x_p, x_1x_1, \ldots, x_1x_p, \\ &\quad x_2x_2, \ldots, x_2x_p, \ldots, x_px_p)^T \qquad (2.21) \\ a &= (a_1, \ldots, a_p, a_{11}, \ldots, a_{1p}, \\ &\quad a_{22}, \ldots, a_{2p}, \ldots, a_{pp})^T. \qquad (2.22) \end{aligned}$$

Similar to kernel methods POLARMAP can implicitly represent the data in a new feature space to improve the transformation [41, 94].

Replacing term f by the respective function we obtain

$$\begin{aligned} \tilde{E} &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T \tilde{x}_i - a^T \tilde{x}_j - \psi_{ij} \right)^2 \qquad (2.23) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T (\tilde{x}_i - \tilde{x}_j) - \psi_{ij} \right)^2. \qquad (2.24) \end{aligned}$$

For a better readability we replace $\tilde{x}_i - \tilde{x}_j$ by $\tilde{x}_{ij}$ and obtain

$$\tilde{E} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T \tilde{x}_{ij} - \psi_{ij} \right)^2. \tag{2.25}$$

The derivative of $\tilde{E}$ w.r.t. a can be easily obtained as

$$\frac{\partial \tilde{E}}{\partial a} = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T \tilde{x}_{ij} - \psi_{ij} \right) \tilde{x}_{ij}. \tag{2.26}$$

Setting equation (2.26) equal to zero to fulfil the necessary condition for a minimum we infer

$$0 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a^T \tilde{x}_{ij} - \psi_{ij} \right) \tilde{x}_{ij} \tag{2.27}$$

which results in a system of linear equations in $a = (a_1, a_2, \ldots, a_p)^T$.

As mentioned already, angles computed by $f(x_i) - f(x_j)$, might be positive or negative, while ψ_{ij} is always positive by definition. Thus, in the case where $a^T \tilde{x}_{ij} < 0$ holds, $\tilde{E}$ might be minimal, but our original objective function E might not be minimal. We discussed this issue already with MDS_{polar} in section 2.3 where an analogous problem arises as well. Hence, replacing $\tilde{x}_{ij}$ by $-\tilde{x}_{ij}$ in this case might lower the error. Consequently, finding the appropriate sign for $\tilde{x}_{ij}$ is a crucial step when minimizing $\tilde{E}$. For usual data sets determining the exact solution for this problem is too expensive regarding computation time. In the following section we describe a greedy strategy that approximates a relaxation of this problem.

2.4.1 A Greedy Algorithm for the Approximation of POLARMAP

Determining the sign for each $\tilde{x}_{ij}$ requires exponential need of computation time in the number of feature vectors. For real-world data sets this is unacceptable. When relaxing the problem in favour to an approximation of the exact solution one can reduce the time complexity down to $O(n \cdot \log n)$. In this section we begin with a very fast greedy algorithm that finds rather poor approximations of the exact solution, which are suitable for initialization purposes for more complex approximation schemes.

In the following we use an $n \times n$-matrix Θ with $\theta_{ij} = 1$ when the sign for $\tilde{x}_{ij}$ is positive and $\theta_{ij} = -1$ when the sign for $\tilde{x}_{ij}$ is negative[4]. As algorithm 3

[4]Implementation aspects to improve efficiency we consider later.

Algorithm 3 Greedy POLARMAP 1

Given the data set $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$
Let $\mathbf{\Psi}_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
Let $\mathbf{\Theta}_{n\times n}$ be a matrix where $\theta_{ij} = 0, \ \forall i, j$
$a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
Compute $\tilde{E}$
repeat
  $\tilde{E}' \leftarrow \tilde{E}$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n **do**
      if $\theta_{ij}a^T\tilde{x}_{ij} < 0$ **then**
        $\theta_{ij} \leftarrow 1 - \theta_{ij}$
      end if
    end for
  end for
  $a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\theta_{ij}a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
  Compute $\tilde{E}$
until $\tilde{E}' \leq \tilde{E}$

shows this greedy algorithm does not change these signs for the respective $\tilde{x}_{ij}$ until $\theta_{ij}a^T\tilde{x}_{ij} < 0$ is satisfied and computes afterwards the updated components of a by solving the revised system of linear equations. Usually, this algorithm converges after a few iterations. This approach is very efficient and simple at the same time.

Algorithm 4 shows another greedy algorithm. Always, when θ_{ij} changes, a will be recomputed immediately and the next iteration starts. Θ changes during one iteration at most in one point – namely θ_{ij}, otherwise the algorithm ends without changing any θ. Thus, the algorithm greedily changes the first θ_{ij}, when condition $\theta_{ij}a^T\tilde{x}_{ij} < 0$ is satisfied. From this it follows that the algorithm only finds a local minimum of $\tilde{E}$, which is the reason why we speak about a relaxation of the problem.

It is advisable to initialize Θ with algorithm 3. Otherwise, too many iterations will be needed until convergence. With algorithm 4 very accurate transformations will be found – indeed computational costs are fairly high.

Algorithm 4 Greedy POLARMAP 2

Given the data set $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$
Let $\mathbf{\Psi}_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
Let $\mathbf{\Theta}_{n\times n}$ be a matrix where $\theta_{ij} = 0, \ \forall i, j$
$a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
Compute $\tilde{E}$
repeat
  $\tilde{E}' \leftarrow \tilde{E}$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n **do**
      if $\theta_{ij}a^T\tilde{x}_{ij} < 0$ **then**
        $\theta_{ij} \leftarrow 1 - \theta_{ij}$
        $a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\theta_{ij}a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
        Compute $\tilde{E}$
        GOTO: check
      end if
    end for
  end for
  LABEL: check
until $\tilde{E}' \leq \tilde{E}$

In the following subsection, we describe a technique that reduces the computation costs drastically.

2.4.2 Generalization of POLARMAP

Although the above greedy algorithm is efficient, for large data sets too many iterations will be needed until convergence. Its time and space complexity are also quadratic in the number of data, so that it is not applicable to larger data sets. In order to evaluate the objective function, all $\tilde{x}_{ij}$- and all ψ_{ij}-values must be computed in advance, causing already the quadratic complexity. The greedy algorithm must also compute many (again quadratic in the number of data) scalar products $a^T\tilde{x}_{ij}$ that contribute not or only little to the quality of the transformation. Thus, if we had a measure to decide

whether an adaptation of θ_{ij} might be target-oriented or not, we could save computation time by skipping the computation of nonessential scalar products.

As a matter of fact, the computation of the error $\tilde{E}$ accounts for most of the computation resources. In the following we will discuss the problem how to reduce computation time due to dropping hopefully dispensable terms. As for MDS_{polar} we provide for POLARMAP a generalization by introducing weights w_{ij} for our objective function $\tilde{E}$, that results in

$$\tilde{E} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \left(a^T \tilde{x}_{ij} - \psi_{ij} \right)^2 . \tag{2.28}$$

Again, we obtain the following system of linear equations after taking partial derivatives of $\tilde{E}$

$$\frac{\partial \tilde{E}}{\partial a} = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \left(a^T \tilde{x}_{ij} - \psi_{ij} \right) \tilde{x}_{ij} \tag{2.29}$$

and setting equation (2.29) to zero to fulfil the necessary condition for a minimum which leads to

$$0 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \left(a^T \tilde{x}_{ij} - \psi_{ij} \right) \tilde{x}_{ij}. \tag{2.30}$$

The introduction of weights opens new ways to define and handle the objective function. We cannot only assign a weight to individual errors, controlling in this way how much influence single errors have on the final result. We can also consider relative instead of absolute errors. For example, choosing $w_{ij} = 1/\psi_{ij}^2$ corresponds to minimizing the relative error, similar as in the case of MDS_{polar}. The difference between the angle in the original space and the corresponding angle in the target space will not account directly to the computation of the parameter a but weighted with w_{ij}. Weights can be chosen in such a way, that only feature vectors, which are similar to a certain degree, will be taken into account, when computing θ_{ij}.

Since our transformation preserves the length of each data vector, it is guaranteed that vectors with a large difference in length will not be mapped

to close points in the plane, even though their angle might not be matched at all. Therefore, we propose to use a small or even zero weight for pairs of data vectors that differ significantly in their length. We skip the discussion of defining appropriate weights here since we explained the use of suitable weighting control functions in section 2.3.5 already in conjunction with MDS_{polar}.

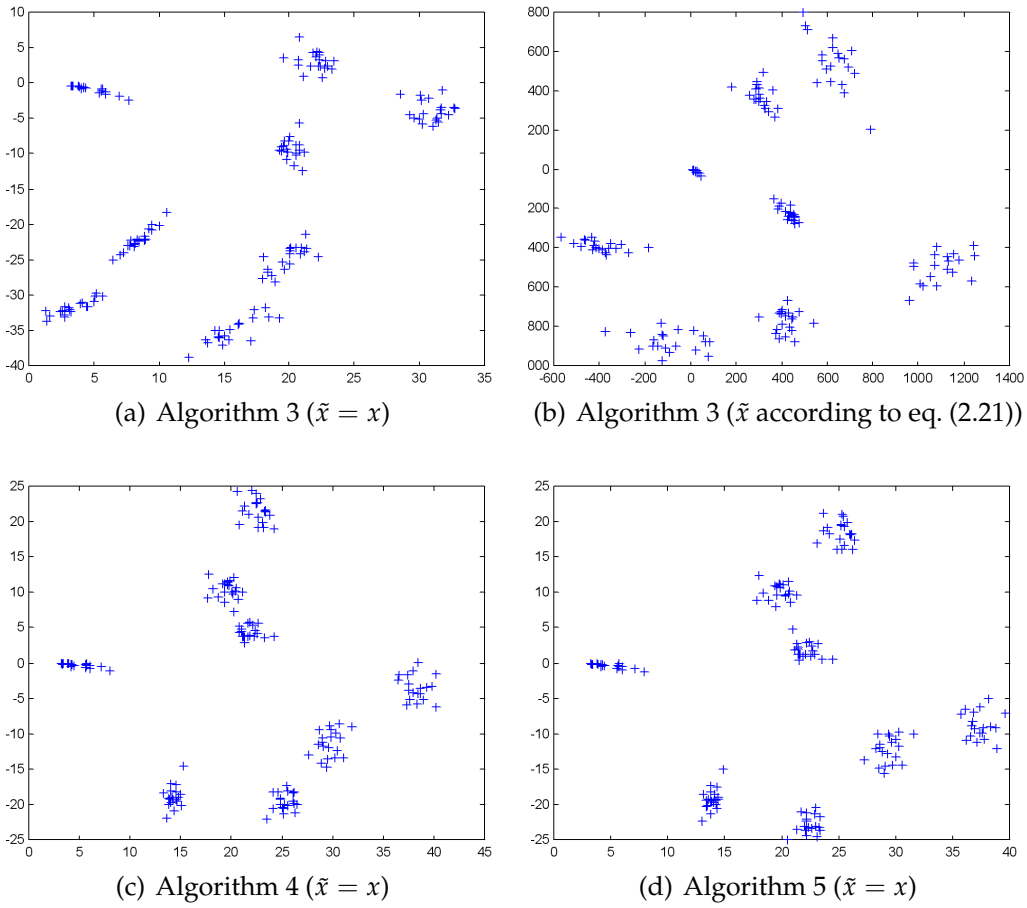
Algorithm 5 is a modification of the previous algorithm considering the mentioned aspects. Besides sorting X, the inner for-loop now contains the condition to terminate the loop if the weighting function indicates that for the given i no further $\tilde{x}_{ij}$ has to be considered.

Note that we reduce computation time drastically, if we choose an appropriate weighting function. Instead of $O(c \cdot n^2)$ with c as number of iterations, we obtain $O(c \cdot n \cdot m)$ where m is the maximum bin size. Since sorting is essential for the binning approach, the costs for sorting have to be added. The bin size for a feature vector x_i refers to the number of non-zero weights w_{ij}. With this binning strategy we do not only reduce the number of pairs $\tilde{x}_{ij}$ to be considered, much more important is the effect on the computation of a and $\tilde{E}$. With algorithm 5 we introduce the array n_i, $i = 1 \ldots n$, that is initialized with $n_i = \min(i + maxbinsize, n)$, $\forall i$. When computing a and $\tilde{E}$, it is no longer necessary to sum up the difference between the target angle and ψ_{ij} for all vectors that are out of the bin, since they be weighted with $w_{ij} = 0$. Of course, if the bin size is only limited by a weighting function that only leads to few weights $w_{ij} = 0$, the gain of computation time tends to zero. Further, a threshold w^* is set which ensures that the bin size can be reduced locally, if, for a given feature vector, the number of similar feature vectors is smaller than the bin size.

The major part in terms of memory usage attributes to the sign matrix Θ since conventional programming languages do not support bit variables or arrays of bits. Using a byte variable to code a single sign would mean eightfold memory wastage. In order to reduce the space complexity it is recommendable to code the sign matrix binary which allows an optimal resource management.

Algorithm 5 Greedy POLARMAP 3

$\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n\}$
Sort X
Set $maxbinsize$
Set w^*
Initialize n_i
Let $\mathbf{\Psi}_{n\times n}$ be a matrix with the pairwise angles ψ_{ij} between all (x_i, x_j)
Let $\mathbf{\Theta}_{n\times n}$ be a matrix where $\theta_{ij} = 0, \ \forall i, j$
$a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n_i} w_{ij}\left(a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
Compute $\tilde{E}$
repeat
  $\tilde{E}' \leftarrow \tilde{E}$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n_i **do**
      $w_{ij} \leftarrow w(l_i, l_j)$
      if $w_{ij} > w^*$ **then**
        $n_i \leftarrow j-1$
        GOTO: check1
      end if
      if $\theta_{ij}a^T\tilde{x}_{ij} < 0$ **then**
        $\theta_{ij} \leftarrow 1 - \theta_{ij}$
        $a \leftarrow solve\left(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n_i} w_{ij}\left(\theta_{ij}a^T\tilde{x}_{ij} - \psi_{ij}\right)\tilde{x}_{ij} = 0\right)$
        Compute $\tilde{E}$
        GOTO: check
      end if
    end for
    LABEL: check1
  end for
  LABEL: check
until $\tilde{E}' \leq \tilde{E}$



(a) Algorithm 3 ($\tilde{x} = x$)

(b) Algorithm 3 ($\tilde{x}$ according to eq. (2.21))

(c) Algorithm 4 ($\tilde{x} = x$)

(d) Algorithm 5 ($\tilde{x} = x$)

Figure 2.6: Results of POLARMAP on the Cube data set

2.4.3 Illustrative Examples

In this section we discuss the results of POLARMAP on two synthetic 3-dimensional data sets that we already showed in figure 2.3. Furthermore, we apply POLARMAP on the well known Iris data set and the Wine data set. A Sammon's mapping of the 4-dimensional Iris data set is shown in figure 2.8 (a). We split this data set into a training data set and a test data set to demonstrate the capability of POLARMAP to generalize. The Wine data set results from a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. A

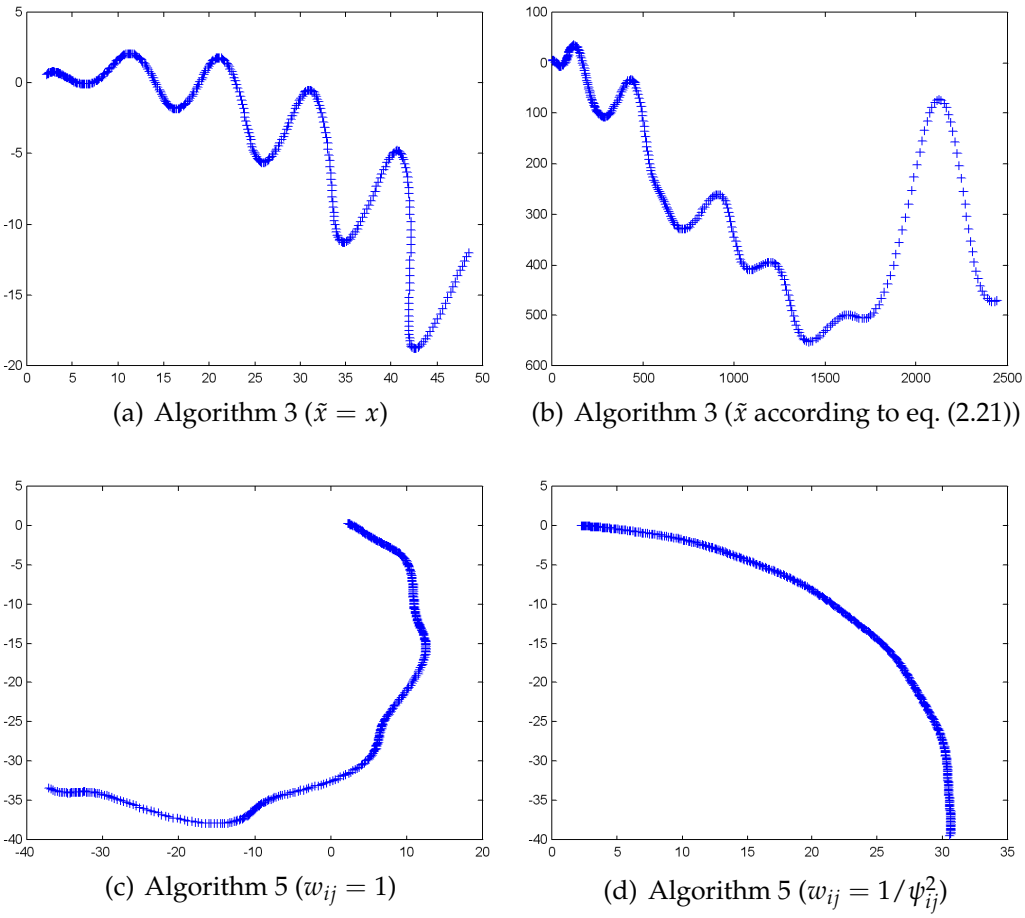
Sammon's mapping of the 13-dimensional Wine data set is shown in figure 2.8 (c).

Our tests have shown, that algorithm 3 is a good initialization for algorithm 4 and 5. Since algorithm 4 and 5 compute the actual coefficients immediately after changing one sign, without initialization, many iterations would be needed for large data sets until convergence. For that reason it is advisable to initialize algorithm 4 and 5 with algorithm 3. Note that algorithm 3 does not have exponential complexity, when we introduce corresponding weights leading to moderate bin sizes. The following transformations result from this procedure.

Figure 2.6 shows some results on the Cube data set. The greedy algorithm 3 converges already after three iterations if using $\tilde{x}$ according to equation (2.20) (see figure 2.6(a)). Similarly, greedy algorithm 3 converges after five iterations, when using $\tilde{x}$ according to equation (2.21) (see figure 2.6(b)). For the relatively simple Cube data set it is not of much importance, to generate additional components for $\tilde{x}$ and additional components a, respectively. Figures 2.6(c) and 2.6(d) result from applying algorithm 4 and algorithm 5, respectively. These transformations are based on the choice $\tilde{x} = x$ for the Cube data set. Obviously, a linear function with three coefficients a is sufficient to map the Cube data set to a 2-dimensional feature space. The eight clusters are clearly separated in the target space, too. Using weights according to algorithm 5 and the following weighting function

$$w(z_a) = \begin{cases} 1 & \text{if } z_a < w^* \\ 0 & \text{otherwise} \end{cases}$$

with $c = \frac{1}{4}n$ leads to the transformation shown in figure 2.6(d). Note, as the histogram depicts (see figure 2.9) roughly four groups of vector lengths are present in the Cube data set. Since the Cube data set contains eight clusters, each one composed of twenty points, the chosen bin size c guarantees that each bin comprehends data from at least two clusters. This has the effect that vectors, whose lengths are similar, will be considered for the adaptation of the angles. The results with algorithm 4 and algorithm 5 are quite similar for the Cube data set, even though algorithm 5 needs less computation time.



(a) Algorithm 3 ($\tilde{x} = x$)

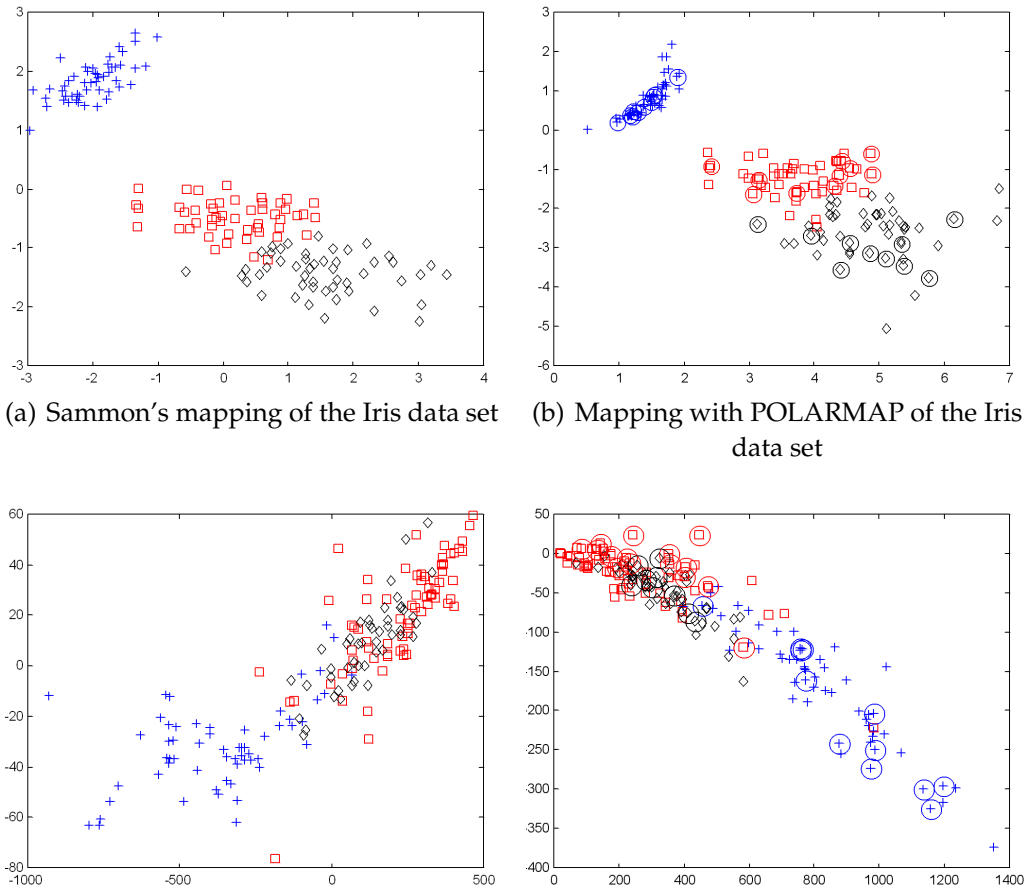
(b) Algorithm 3 ($\tilde{x}$ according to eq. (2.21))

(c) Algorithm 5 ($w_{ij} = 1$)

(d) Algorithm 5 ($w_{ij} = 1/\psi_{ij}^2$)

Figure 2.7: Results of POLARMAP on the Coil data set

The results for the Coil data set are shown in figure 2.7. Again, figure 2.7(a) results from algorithm 3 with $\tilde{x} = x$ and figure 2.7(b) results from $\tilde{x}$ according to equation (2.21). The results for both representations of the data set are similar regarding a majority of the characteristics. Algorithm 3 converges after few iterations for both data sets. Figures 2.7(c) and 2.7(d) show the results of algorithm 5 on the data set ($\tilde{x} = x$) without initializing Θ other than 1. Figure 2.7(c) results from using a bin size of 10. Figure 2.7(d) results from using $w_{ij} = 1/\psi_{ij}^2$ for all w_{ij} inside the bin of size 10. Choosing maximum bin size 10 leads to the fact that no sign will be changed and thus the algorithm stops after one iteration. Designing the weighting function such that a larger bin



(a) Sammon's mapping of the Iris data set

(b) Mapping with POLARMAP of the Iris data set

(c) Sammon's mapping of the Wine data set

(d) Mapping with POLARMAP of the Wine data set

Figure 2.8: Results of Sammon's mapping and POLARMAP on the Iris data set and the Wine data set

size has to be taken into account, one can observe that already after only three signs have changed, the transformation gets the major characteristics as the one in figure 2.7(a). Even if this transformation is very simple, some requirements, e.g. preserving distances between feature vectors approximately, are fulfilled.

Since a function is learned by POLARMAP it becomes possible to map new vectors to the target space. To demonstrate the power of POLARMAP in this regard, we applied it on the well known Iris data set. Figure 2.8(a)

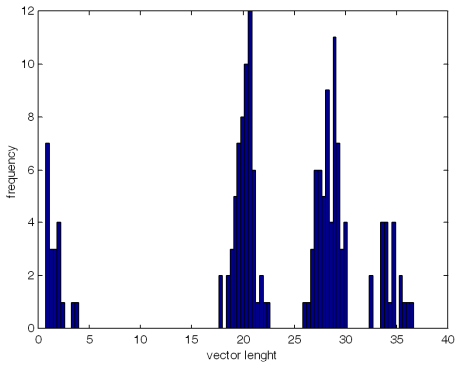


Figure 2.9: Histogram of vector lengths of the Cube data set

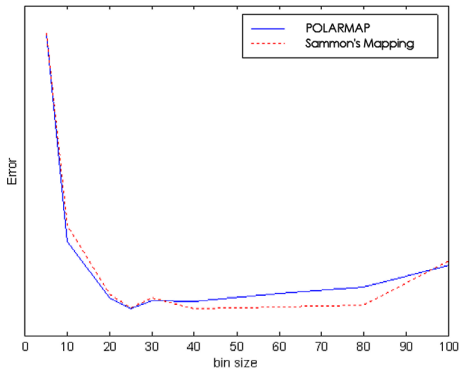


Figure 2.10: The effect of the bin size on POLARMAP and Sammon's mapping

shows the Sammon's mapping of the Iris data set. The different classes are represented by different symbols (class 1: $+$, class 2: $\square$, class 3: $\diamond$). Figure 2.8(b) shows the transformation with POLARMAP. For this example, the Iris data set is split into a training data set and a test data set. The training data set consists of 80% of each class. This part of the data is used to learn the desired coefficients. The test data set that contains the remaining 20% of the data, is mapped to the target space by means of the learned function. The mapping of the training data set is plotted with the different symbols again, each for the corresponding class. The mapped feature vectors of the test data set are additionally marked with a circle around the corresponding symbol. As the figure shows, the learned function maps the feature vectors according to our intuition.

Figure 2.8(c) shows the Sammon's mapping of the Wine data set. The three classes are marked with different symbols again. Based on the Sammon's mapping, the three classes cannot be separated linearly. Notably class 2 and class 3 cannot be distinguished. The transformation of the Wine data set with POLARMAP is shown in figure 2.8(d). Both transformations are similar regarding the scattering of the different classes. The mapping of the test data (marked with a circle around the respective symbol) meets the expectations from the mapping of the training data set.

Figure 2.10 shows the effect of the bin size on the transformation accuracy

according to the POLARMAP criterion (solid line) and the Sammon criterion (dashed line) on the Wine data set[5]. Both measures indicate a better mapping with increasing bin size at the beginning. This is what we expect indeed. For larger bin sizes the error is increasing slightly again. The probability to get stuck in a local minimum seems to increase with larger bin sizes. As the figure reveals as well, the error is not decreasing linearly. Thus, using the binning technique, the user has to make the compromise between transformation quality and computation/space complexity. In many cases it may be sufficient to use a small bin size to get an overall view of the data. As for the Wine data set, the error can be reduced drastically, investing resources in the consideration of a higher bin size.

2.4.4 Visualizing Weather Data with POLARMAP

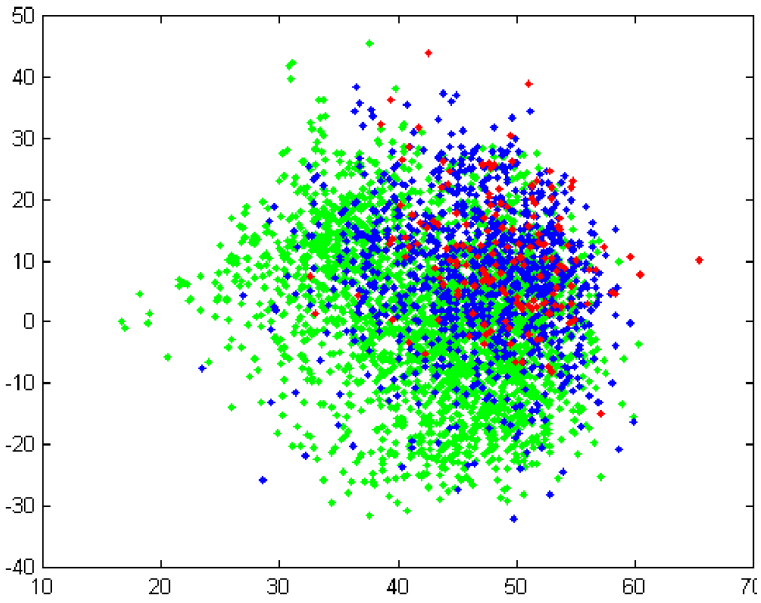
Recently, we have published a study concerning the prediction of aircraft flight durations [61, 78]. The study is based on two combined data sets, concerning the weather conditions and the traffic, respectively, at Frankfurt Airport. A brief description of the data sets was given in section 1.3.

POLARMAP is applied to the data to visualize the relationships between flight duration and weather factors. An earlier study has shown that the weather can be easily divided into cloudy and non-cloudy weather since the data form two distinct clusters that can be separated almost linearly [78]. For the purpose of visualization this procedure is followed and both subsets of data are transformed separately. For illustration purposes, we discuss in this work just the non-cloudy weather. In the mentioned study, a strong effect of the traffic on the travel time could be determined. Therefore, this section shows visualization results of the weather data, both, including the traffic attribute and excluding it.

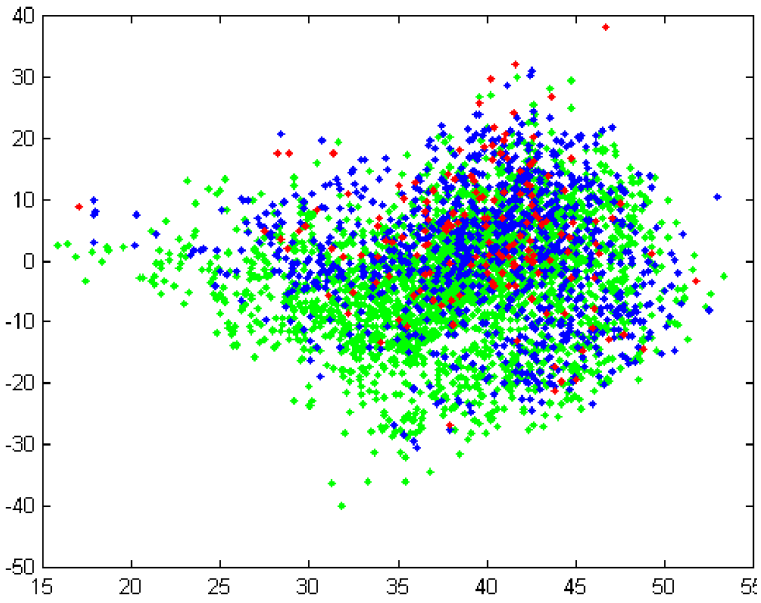
Figure 2.11(a) shows a mapping of the non-cloudy weather including the

[5]The error in the figure is without a unit. Actually, both measures cannot be used directly for comparison of the two methods since both optimization criteria are quite different. Of course, the Sammon stress for a POLARMAP transformation is quite higher than for Sammon's mapping and vice versa. Thus, the normalized error in the figure only reflects the behaviour of both techniques varying the bin size.

traffic attribute using POLARMAP. The colouring that is used here to visualize the distribution of different flight duration classes has been already discussed in section 2.3.7. As for the mapping with MDS_{polar}, the figure shows clearly that the three classes overlap to some degree. Whereas the green class (flights with short flight durations) is spread all over the feature space, the blue class (flights with medium flight durations) and the red class (flights with long flight durations) take the upper right region of the feature space mainly. This effect is mainly due to the traffic information. Figure 2.11(b) shows the transformation of the same data set excluding the traffic information. This figure reveals that, without traffic information, the different classes are even harder to distinguish.



(a) Transformation with POLARMAP of the non-cloudy weather data including traffic information



(b) Transformation with POLARMAP of the non-cloudy weather data excluding traffic information

Figure 2.11: Mapping of the weather data set with POLARMAP

2.5 Density-Based Mappings

The idea of density-based visualization is to reflect density variations of high-dimensional data sets. We have presented this approach in [88]. In the following we formalize the problem of density preservation by means of an objective function that can be minimized through a gradient descent technique.

For each data object in the original data space a multivariate Gaussian distribution is defined that represents a data point's potential energy. When adding those single potentials we get a sort of multidimensional potential mountains. Summits of the mountains can be found where many data objects are located. Accordingly, valleys can be found in areas of low data density.

Similarly, one can reproduce the mountains in the low-dimensional feature space (usually two or tree dimensions). For this purpose each data object of the original space will be placed in the projection space. Over every single data point a potential (in form of a two- or three-dimensional Gaussian distribution) will be applied. The criterion for the mapping is that the potentials in the original space coincide as good as possible with the potentials at the corresponding points in the target space.

Given the data set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ we seek for the mapped data set $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^k$ with $k = 2$ or $k = 3$ with the following potential for x_i:

$$f_i(x) \;=\; \frac{1}{c} \exp\left(-\frac{1}{2} \sum_{t=1}^{p} \left(\frac{x^{(t)} - x_i^{(t)}}{\sigma} \right)^2 \right) \tag{2.31}$$

with

$$c \;=\; \frac{1}{\sigma^p \sqrt{(2\pi)^p}}.$$

By $x^{(t)}$ and $x_i^{(t)}$ we denote the t^{th} attribute of data object x and x_i, respectively. Function f_i simply describes the density of a p-dimensional Gaussian distribution with mean value x_i and variance σ^2 in each dimension. The parameter σ must be fixed for the entire procedure. If σ is rather small, then the

potentials do rarely overlap. For very large σ the potential landscape will be blurred completely with little variance in height.

Therefore, it is useful to define σ according to the diameter d of the data space, the average distance between data points, the number n of data and the dimensionality p. A straight forward approach would be to assume that the data is uniformly distributed in a hyper-cube or hyper-sphere. In this case the potentials would have approximately the same height. Of course, this assumption is fairly theoretical. In practice mountains will be formed due to the heterogeneous structure of the data. However, under this assumption the average density can be computed and the potentials on and between data points can be determined. The larger the variance σ^2, the smaller the difference in the potentials. For small data sets the density is low and therefore a larger σ should be chosen.

Similar to Sammon's mapping we seek the projected data points $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^k$. Over each data point we apply a potential (in this case a k-dimensional Gaussian distribution) as for the original space:

$$g_i(y) = \frac{1}{\tilde{c}} \exp\left(-\frac{1}{2} \sum_{t=1}^{k} \left(\frac{y^{(t)} - y_i^{(t)}}{\tilde{\sigma}} \right)^2 \right) \tag{2.32}$$

with

$$\tilde{c} = \frac{1}{\tilde{\sigma}^k \sqrt{(2\pi)^k}}.$$

Then the objective is to place the feature vectors such that the potentials coincide at least in these points with those in the original space. Note, $\tilde{\sigma}$ should be chosen similarly to σ. In the ideal case we have approximately the same diameter d in the target space, too. However, the area (or the volume) of the target space will be much smaller compared to the hyper volume of the original space ($k \ll p$). This means that the density in the target space is also higher for the same size of the data set. Thus, $\tilde{\sigma}$ should be chosen smaller than σ. Still the potentials in the target space might not match the potentials in the original space yet. It should be assured that the maximum height of the single potentials in the original space and in the target space match, i.e.

the respective maxima of the Gaussian distributions should be:

$$f_i(x_i) \approx g_i(y_i).$$

Since normally this will not be the case we introduce a constant a:

$$a \cdot f_i(x_i) = g_i(y_i)$$

which can be derived from equations (2.31) and (2.32):

$$a = \frac{\sigma^p}{\tilde{\sigma}^k}\sqrt{(2\pi)^{p-k}}.$$

Now we can formulate our objective function. The accumulated modified potential in the original space at x_i is

$$\sum_{j=1}^{n} a \cdot f_j(x_i)$$

and in the target space at y_i

$$\sum_{j=1}^{n} g_j(y_i).$$

In the ideal case, both potentials should be equal. Hence, we define the objective function as follows:

$$\begin{aligned} E_{density} &= \sum_{i=1}^{n}\left(\sum_{j=1}^{n} g_j(y_i) - \sum_{j=1}^{n} a \cdot f_j(x_i)\right)^2 \\ &= \sum_{i=1}^{n}\left(\sum_{j=1}^{n}\left(g_j(y_i) - a \cdot f_j(x_i)\right)\right)^2. \end{aligned} \tag{2.33}$$

Now, we only have to determine the gradient for each component s:

$$\frac{\partial E_{density}}{\partial y_{ls}} = 2\sum_{i=1}^{n}\sum_{j=1}^{n}\left(g_j(y_i) - a \cdot f_j(x_i)\right) \cdot \frac{\partial}{\partial y_{ls}} g_j(y_i). \tag{2.34}$$

$\frac{\partial}{\partial y_{ls}} g_j(y_i)$ is only zero when we have $l = i$ or $l = j$. For both cases we derive from equation (2.34):

$$\frac{\partial}{\partial y_{ls}} g_l(y_i) = \frac{1}{\tilde{c}} \exp\left(-\frac{1}{2}\sum_{t=1}^{k}\left(\frac{y_i^{(t)} - y_l^{(t)}}{\tilde{\sigma}}\right)^2\right) \cdot \frac{y_i^{(s)} - y_l^{(s)}}{\tilde{\sigma}} \tag{2.35}$$

$$\frac{\partial}{\partial y_{ls}} g_j(y_l) \;=\; -\frac{1}{\tilde{c}} \exp\left(-\frac{1}{2} \sum_{t=1}^{k} \left(\frac{y_l^{(t)} - y_j^{(t)}}{\tilde{\sigma}} \right)^2 \right) \cdot \frac{y_l^{(s)} - y_j^{(s)}}{\tilde{\sigma}}. \qquad (2.36)$$

It can be easily seen that for $i = j = l$ we have $\frac{\partial}{\partial y_{ls}} g_j(y_i) = 0$. Finally we obtain for the gradient:

$$\begin{aligned} &\frac{\partial E_{density}}{\partial y_{ls}} \\ &= \frac{2}{\tilde{c}} \sum_{i=1}^{n} \left((g_l(y_i) - a \cdot f_l(x_i)) \cdot \exp\left(-\frac{1}{2} \sum_{t=1}^{k} \left(\frac{y_i^{(t)} - y_l^{(t)}}{\tilde{\sigma}} \right)^2 \right) \cdot \frac{y_i^{(s)} - y_l^{(s)}}{\tilde{\sigma}} \right. \\ &- \left. (g_i(y_l) - a \cdot f_i(x_l)) \cdot \exp\left(-\frac{1}{2} \sum_{t=1}^{k} \left(\frac{y_l^{(t)} - y_i^{(t)}}{\tilde{\sigma}} \right)^2 \right) \cdot \frac{y_l^{(s)} - y_i^{(s)}}{\tilde{\sigma}} \right). \end{aligned} \qquad (2.37)$$

Combining the Sammon gradient E_{sammon} (see equation 2.2) and the density gradient $E_{density}$ through linear combination we finally obtain:

$$E = \alpha \frac{\partial E_{sammon}}{\partial y_l} + \beta \frac{\partial E_{density}}{\partial y_l}. \qquad (2.38)$$

The parameters α and β can be considered as learning rates or weights to control the impact of the respective mapping strategy. Thus, higher weights α for the Sammon gradient favour distance-based mappings and larger values β for the density gradient favour the density approach.

2.5.1 Illustrative Examples

In this section we will discuss some results of the proposed technique on the Cube data set and on the Wine data set. Figure 2.13 shows a Sammon's mapping of the Cube data set. The eight data clusters are well reflected by the mapping. The transformation with the density-based approach, setting $\alpha = 0$ and thusly optimizing the density aspect exclusively, leads to the mapping visualized in figure 2.14. It is surprising that already the density aspect in the optimization is sufficient in this example to reflect the structure of the data set. Applying a linear combination of both, the Sammon gradient and the density gradient, we obtain the mapping depicted in figure 2.15.

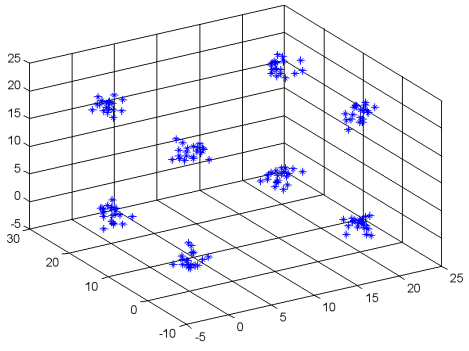


Figure 2.12: Cube data set

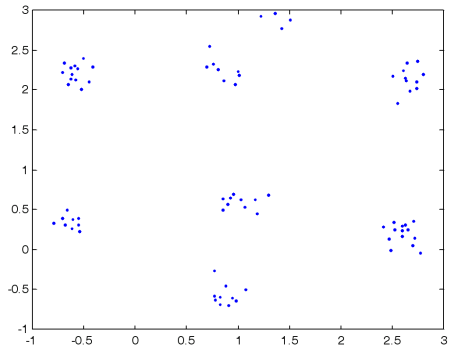


Figure 2.13: Sammon's mapping of the Cube data set

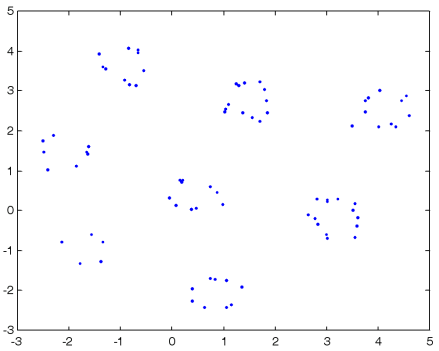


Figure 2.14: Density-based mapping of the Cube data set

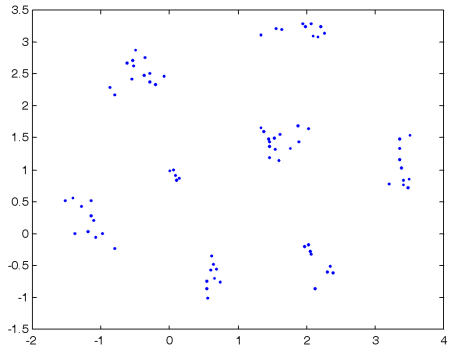


Figure 2.15: Mapping of the Cube data set (distance-based and density-based)

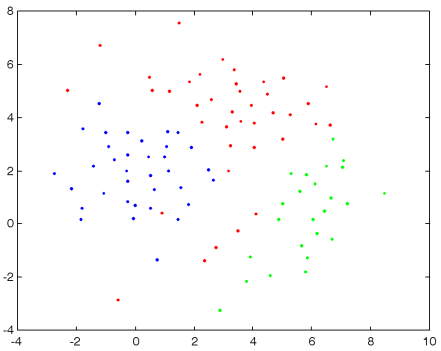


Figure 2.16: Sammon's mapping of the Wine data set

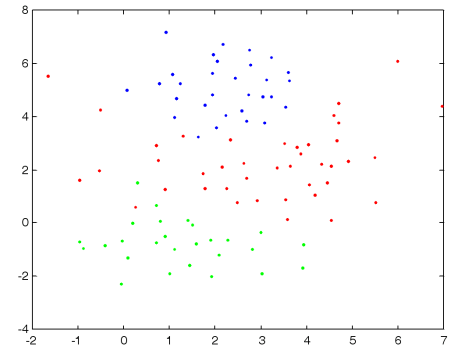


Figure 2.17: Density-based mapping of the Wine data set

Whereas the distance-based approach seams to favour the preservation of the inter-cluster structure, the linear combination of distance and density aspects gives a better overall impression of the data set. Figures 2.16 and 2.17 show transformations of the Wine data set with Sammon's mapping and with the density-based approach, respectively. Both transformations show similar characteristics.

Based on the empirical tests we cannot constitute that the density-based approach is superior to the distance-based approach. Indeed, the computational complexity per iteration of the density-based approach is rather higher since the density gradient has to be computed additionally. Our tests have shown that the number of iterations can be reduced with density preservation.

2.6 Navigation Through High-Dimensional Data

In this section we discuss a visualization technique that maps a high-dimensional data set onto a plane without optimizing any objective function or using an iteration scheme. Thus, our method is linear with the number of objects.

The method we propose in this work is to map the data set into a 2-dimensional feature space by means of preserving angles to a certain reference vector. Initially, a reference vector r can be chosen at random out of the original data set or it can be an appropriately generated vector, e.g. the centre vector of the data set in the original space. Not only the angle φ_{rj} to the reference vector will be preserved but also the length of each feature vector x_j. Thus, for the target vector $\hat{x}_j$ we obtain the following simple equation for the transformation

$$\hat{x}_j = \begin{pmatrix} \cos\varphi_{rj} \, |x_j| \\ \sin\varphi_{rj} \, |x_j| \end{pmatrix}.$$

Of course, when choosing the reference vector randomly, an adequate transformation can be only obtained by chance. Therefore, it would be advisable to choose the reference vector by means of an appropriate heuristic. We suggest in this work two different approaches. If there is no further information

about the data available transformations with any reference vector could be of interest. Hence, a dynamic visualization that shows transformations of the data set by means of an animation could reveal interesting data structures. Such a dynamic visualization would only be helpful if consecutive transformations are derived from slightly changing reference vectors. Otherwise, the perspective of the transformation would always change drastically, since the reference vector jumps from anywhere in the feature space to anywhere else without any plan. The only problem is how similarity can be ascertained. An ad hoc approach would be to sort the feature vectors by means of their length. A sorting like this reflects at least that feature vectors with a short length are quite different from those with a great length. The fact that feature vectors with a similar length can have a far distance to each other will not be recovered by sorting. Clustering could be another technique to choose the sequence of reference vectors according to their similarity. For a sequence of visualizations one would choose feature vectors from the original data set to act as reference vectors beginning with the one that is the closest to the cluster prototype vector going stepwise to the border of the cluster. If all feature vectors of a certain cluster have served as reference vector, the next cluster will be processed analogously. For large data sets it might be sufficient to use the prototype vectors of a clustering partition for the dynamic visualization.

2.6.1 Illustrative Examples

Before we apply our visualization method on a benchmark data set, we demonstrate some qualities on an artificial data set. Figure 2.18(a) shows again the Cube data set, which was already introduced in section 2.3.6. Thus, the Cube data set contains eight well separated clusters. Figures 2.18(b)-2.18(d) show some transformations of the Cube data set. As these figures show, if the reference vector is randomly chosen, quite different transformations will be obtained. Figure 2.18(b) shows a transformation that reveals four clusters. Obviously, some clusters are overlapped by others. Labelling the clusters and using different symbols or colours for the plot would visualize this effect. This is what figure 2.18(c) shows. Each of the two clusters in the middle

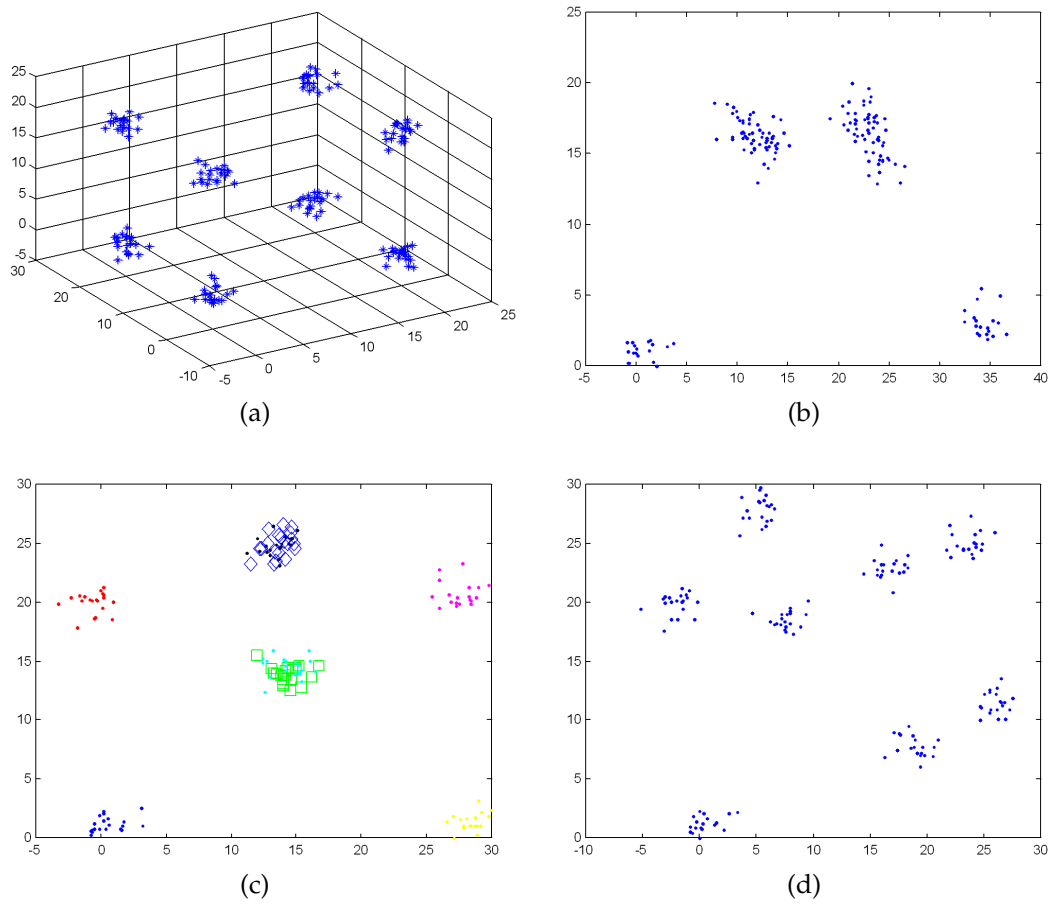


Figure 2.18: Some transformations of the Cube data set with data navigator

represent again two clusters of the original data set. To distinguish the different origin of these points they are drawn by diamonds (◇) and squares (□). Finally, figure 2.18(d) shows a transformation that most suitably preserves the original cluster structures. The differences in the three transformations originate from the fact that – depending on the choice of the reference vector – some feature vectors have approximately the same angle to the reference vector and similar length. Even if these feature vectors have large distances to each other in the original space, they will be placed to similar regions in the target space.

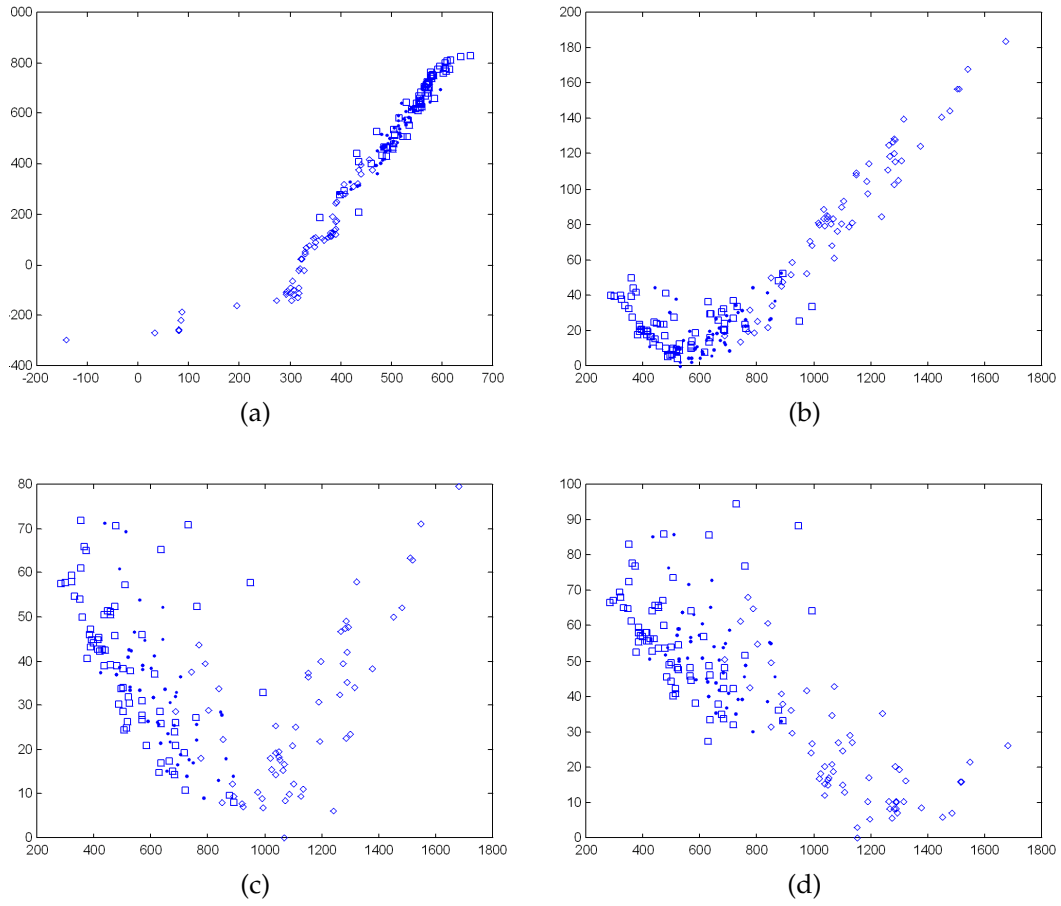


Figure 2.19: Some transformations of the Wine data set with data navigator

Figure 2.19 shows some transformations of the well known Wine data set that was introduced in section 2.4.3 already . Figure 2.19(a) shows a Sammon's mapping of the Wine data set. Figures 2.19(b)-(d) show some transformations with the proposed visualization technique. As the figures reveal, both techniques – Sammon's mapping and the proposed method – lead to similar results regarding some aspects. Obviously, the class symbolized by the diamond (◇), can be separated much better from the rest of the data as the other two classes. The class symbolized by a point and the one symbolized by the square (□) overlap in both transformations. Different from Sammon's mapping, which needs hundreds of iterations for one transformation, our approach runs only once through the data. This permits to provide interac-

tive and even dynamic visualizations. By interaction vectors of interest can be chosen as reference vector.

Of course, a simple transformation without any optimization heuristics often yields fairly poor results. In contrast to many other techniques aiming at dimension reduction, see e.g. [54, 60, 62], the proposed technique is very fast and can thusly give interesting insights to the data.

3 Fuzzy Clustering and Cluster Visualization

Cluster analysis divides data into groups (clusters) such that similar data objects belong to the same cluster and dissimilar data objects to different clusters. The resulting data partition improves data understanding and reveals its internal structure. Partitional clustering algorithms divide a data set into clusters or classes, where similar data objects are assigned to the same cluster whereas dissimilar data objects should belong to different clusters. In real applications there is very often no sharp boundary between clusters so that fuzzy clustering is often better suited for the data. Membership degrees between zero and one are used in fuzzy clustering instead of crisp assignments of the data to clusters. The most prominent fuzzy clustering algorithm is fuzzy c-means, a fuzzification of k-means [7].

3.1 Fuzzy c-means Clustering

Many fuzzy clustering algorithms aim at minimizing an objective function that describes the sum of weighted distances d_{ij} between c prototype vectors v_i and n feature vectors x_j of the feature space $\mathbb{R}^p$:

$$J = \sum_{i=1}^{c} \sum_{j=1}^{n} (u_{ij})^m d_{ij}. \tag{3.1}$$

With the fuzzifier $m \in (1, \infty]$ one can determine how much the clusters overlap. While high values for m lead to overlapping clustering solutions, small values, m tending to 1, lead to rather crisp partitions. In order to avoid the trivial solution assigning no data to any cluster by setting all u_{ij} to zero and

avoiding empty clusters, the following constraints are required:

$$u_{ij} \in [0,1] \qquad 1 \leq i \leq c,\ 1 \leq j \leq n \tag{3.2}$$

$$\sum_{i=1}^{c} u_{ij} = 1 \qquad 1 \leq j \leq n \tag{3.3}$$

$$0 < \sum_{j=1}^{n} u_{ij} < n \qquad 1 \leq i \leq c. \tag{3.4}$$

When the squared Euclidian norm

$$d_{ij} = d^2(v_i, x_j) = (x_j - v_i)^T (x_j - v_i)$$

is used as distance measure for distances between prototype vectors v_i and feature vectors x_j, the fuzzy clustering algorithm is called *fuzzy c-means algorithm*. Modifications of the fuzzy c-means algorithm (FCM) by means of the distance measure, e.g. by using the Mahalanobis distance [63], allow the algorithm to adapt different cluster shapes. Two common representatives applying this modification are the algorithms of Gustafson-Kessel (GK) and Gath-Geva (GG) [33, 36]. The minimization of the functional (3.1) represents a nonlinear optimization problem that is usually solved by means of Lagrange multipliers, applying an alternating optimization scheme [6]. This optimization scheme considers alternatingly one of the parameter sets, either the membership degrees

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{\frac{1}{m-1}}} \tag{3.5}$$

or the prototype parameters

$$v_i = \frac{\sum_{j=1}^{n} (u_{ij})^m x_j}{\sum_{j=1}^{n} (u_{ij})^m} \tag{3.6}$$

as fixed, while the other parameter set is optimized according to equations (3.5) and (3.6), respectively, until the algorithm finally converges. Nevertheless, the alternating optimization scheme can lead to a local optimum. Therefore, it is advisable to execute several runs of FCM to ascertain a reliable partition. With the Euclidian distance measure the fuzzy c-means algorithm finds approximately equally sized (hyper)-spherical clusters.

Algorithm 6 Fuzzy c-means clustering

Given the data set $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^p$
Set number of prototypes $c \in \{2, \ldots, n-1\}$
Set fuzzifier m
Set error E
Set maximum number of iterations t_{max}
Set iteration counter $t = 0$
Initialize partition matrix U
repeat
  Calculate prototype vectors v_i
  Update partition matrix U^t
  $t = t + 1$
until $\|U^{(t)} - U^{(t-1)}\| \leq E$ **or** $t \geq t_{max}$
Output partition matrix U
Output prototype vectors V

3.2 Possibilistic Clustering

For FCM probabilistic membership degrees are used. The sum of all membership degrees for each datum equals 1. A disadvantage of probabilistic fuzzy clustering is that values of memberships do not express how typical a datum is regarding a certain cluster.

For possibilistic c-means (PCM), a variation of FCM, condition (3.3) will be dropped. Whereas the sum of membership degrees to overlapping clusters thusly can be > 1, the sum of membership degrees to outliers can be < 1. To avoid the trivial solution of the optimization problem setting all membership degrees to zero, the objective function is modified to

$$J = \sum_{i=1}^{c} \sum_{j=1}^{n} (u_{ij})^m d_{ij} + \sum_{i=1}^{c} \eta_i \sum_{j=1}^{n} (1 - u_{ij})^m. \tag{3.7}$$

The first term provides minimization of the distances between feature vectors and prototype vectors as for probabilistic clustering. The second term is used to maximize the membership degrees u_{ij}. η_i specifies the distance where the membership degree to cluster i has the value 0.5, in other words the cluster's

expected size. According to equation (3.7) the following formula describes the calculation specification for the membership degrees

$$u_{ij} = \frac{1}{1 + \left(\frac{d_{ij}}{\eta_i}\right)^{\frac{1}{m-1}}}. \tag{3.8}$$

The computation of the prototype vectors is carried out as for FCM. In [57] it is proposed to determine η_i as follows

$$\eta_i = K \frac{\sum_{j=1}^{n} u_{ij}^m d_{ij}}{\sum_{j=1}^{n} u_{ij}}. \tag{3.9}$$

Usually, K will be set to 1. It is advised to estimate η_i based on a preceding probabilistic partitioning. In contrast to FCM where the characteristic of partitioning is emphasized and expressed by means of the partition matrix, membership degrees determined by PCM describe how typical a feature vector is regarding a certain cluster. Figure 3.1 illustrates this property on a simple example. The figure shows the centre vectors of an FCM-clustering (blue) and the centre vectors of a PCM-clustering (green) as well. The membership of two interesting points exposes the difference. The lower left point is quite distant from any cluster but yields a high membership degree to the left cluster for the FCM case. Although, this meets not our expectation regarding natural membership to the left cluster, a forced decision whether the point belongs to the left cluster or to the right cluster leads to the noted membership degrees, since FCM operates exactly like this. We get more intuitive membership degrees using PCM. Since both points of interest have large distances to any prototype vector, their membership degrees to any of them is fairly low.

PCM widely depends on initialization. It is advisable to initialize prototypes and membership degrees with FCM. Different from probabilistic clustering single prototypes cannot be influenced by other prototypes. It may occur that some identical clusters will be found with PCM. A common approach is to initialize PCM with the initial value for η_i obtained from FCM and then to re-calculate η_i followed by few iterations with the new value of η_i.

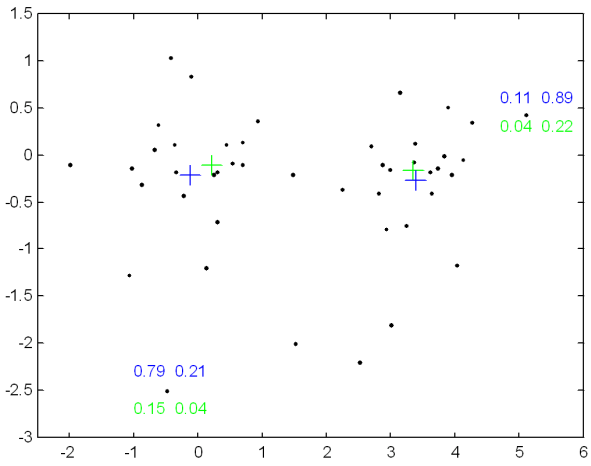


Figure 3.1: An illustrative example of a 2-cluster-partition with FCM and PCM.

PCM is suitable for outlier detection when the data set comprehends compact and separable clusters. Initialization is a crucial step, anyhow. Due to its sensitivity to initialization including the chosen number of prototypes and the value of η_i, it often occurs that too many feature vectors will be regarded as outliers. Improvements of PCM that overcome the shortcoming of finding identical clusters are given in [58, 72, 102].

3.3 Fuzzy Clustering with Outliers

This section briefly describes a modified objective function with an additional weighting factor for each datum [47]. The aim of this approach is not only to assign fuzzy membership degrees to the data points, but also to determine a kind of representativeness of each datum. The approach is designed to enable the expert to determine and separate the critical data from the whole sample data so that the influence of outliers will be reduced. The objective function of outlier clustering

$$J_{outlier} = \sum_{i=1}^{c} \sum_{j=1}^{n} \frac{1}{w_j^r} (u_{ij})^m d_{ij} \tag{3.10}$$

differs only in one point – the introduction of the weighting factor w_j whose influence can be controlled with the constant parameter r. Considering the constraint

$$\sum_{j=1}^{n} w_j = w \tag{3.11}$$

where w is a constant value leads to the following equation for the weighting parameter

$$w_j = \frac{(\sum_{i=1}^{c}(u_{ij})^m d_{ij})^{\frac{1}{r+1}}}{\sum_{l=1}^{n}(\sum_{i=1}^{c}(u_{il})^m d_{il})^{\frac{1}{r+1}}} \cdot w. \tag{3.12}$$

The membership degrees will be calculated using

$$\tilde{u}_{ij}^m = \frac{u_{ij}^m}{w_j^r}. \tag{3.13}$$

The calculation of the prototypes does not differ from the fuzzy c-means scheme.

The objective of outlier clustering is to assign small weighting factors w_k to feature vectors fitting well to at least one cluster, increasing their influence on the clustering result in this way. Outliers are naturally defined as being points having a large distance to all data clusters or being equally shared among many clusters. Outlying points get a large weight, so that they have a small influence on the clustering result. Parameter r influences the clustering procedure as follows: for $r \to \infty$ all $w_j \to \frac{w}{n}$ and no outlier treatment is done. For $r \to 0$ the weighting influence reaches its maximum.

3.4 Noise Clustering

Fuzzy clustering with the fuzzy c-means algorithm allows, based on the membership degrees u_{ij}, the estimation of the degree of assignment of a feature vector x_j to a prototype vector v_i. Since the sum of all membership degrees of a feature vector equals one, according to equation (3.3), even outliers can achieve high membership degrees as we have seen in figure 3.1. Small

membership degrees occur always due to border regions between two or more clusters. The treatment of noisy data is often motivated by the fact that measurements are naturally imperfect. The performance of noise sensitive clustering methods is strongly affected since the presence of outliers impairs the assignment of the representative data [15].

The idea of noise clustering (NC) is based on the introduction of an additional cluster that is supposed to contain all outliers [18, 19, 20]. Feature vectors that are about the noise distance δ or further away from any other prototype vector get high membership degrees to this noise cluster. Hence, the prototype for the noise cluster has no parameters. Let v_c be the noise prototype and x_j the feature vector. Then the noise prototype is such that the distance d_{cj}, the distance from feature vector x_j to v_c, is the fixed constant value

$$d_{cj} = \delta^2, \quad \forall j.$$

The remaining $c - 1$ clusters are assumed to be the good clusters in the data set. The prototype vectors of these clusters are optimized in the same way as mentioned in equation (3.6). The membership degrees are also adapted as described in equation (3.5). As mentioned above, the distance to the virtual prototype is always δ. The only problem is the specification of δ. If δ is chosen too small, too many points will get classified as noise, while a large δ leads to small membership degrees to the noise cluster, which means that noise data are not identified and have a strong influence on the prototypes of the regular clusters.

3.5 Noise Clustering Based Outlier Detection

The specification of the noise distance depends on several factors, e.g. the maximum percentage of the data set to be classified as noise [48], the distance measure, the number of assumed clusters and the expansion of the feature space. The noise distance proposed in [17] is a simplified statistical average

over the non-weighted distances of all feature vectors to all prototype vectors

$$\delta^2 = \lambda \frac{\sum_{i=1}^{c-1} \sum_{j=1}^{n} d_{ij}}{n(c-1)}$$

where λ is the value of the multiplier used to obtain δ from the average of distances. As mentioned above one can show that in this way δ suffers from the fact that with an increasing number of prototypes δ takes relatively high values. As a consequence, the placing of the prototypes will be affected by the outliers, which we intended to avoid.

We proposed in [79, 81] a noise distance which depends primarily on the number of prototypes used for the clustering process and the expansion of the feature space. Under the constraint of the preservation of the hypervolume of the feature space, we choose for δ a value which corresponds to the cluster radius of the hyperspherical cluster. The cluster radius, so δ, will be chosen such that the sum of the hypervolumes of the $c-1$ good clusters with approximately same size, equals the hypervolume of the feature space. A uniformly distributed feature space would not have any outliers in this case. Consequently, if there are regions of high density, some prototypes will be attracted to these regions. Feature vectors which are located a larger distance away from any other prototype vector get high membership degrees to the noise cluster.

So the first step is to estimate the hypervolume of the feature space. A simple solution for this is shown in Figure 3.2(a). By means of the data set's extreme feature vectors the area of the resulting rectangle, or more generally the hypervolume V of the cuboid in an n-dimensional feature space, can be easily computed. A closer approximation of the hypervolume V can be achieved by subdividing the feature space into smaller pieces and summing up the single volumes of the respective hyperboxes. Figures 3.2(b) and 3.2(c) show two naive partitions of the feature space. Note that such a grid should subdivide the feature space into hyperboxes of approximately the same size, since the fuzzy c-means algorithm searches for approximately equally sized clusters.

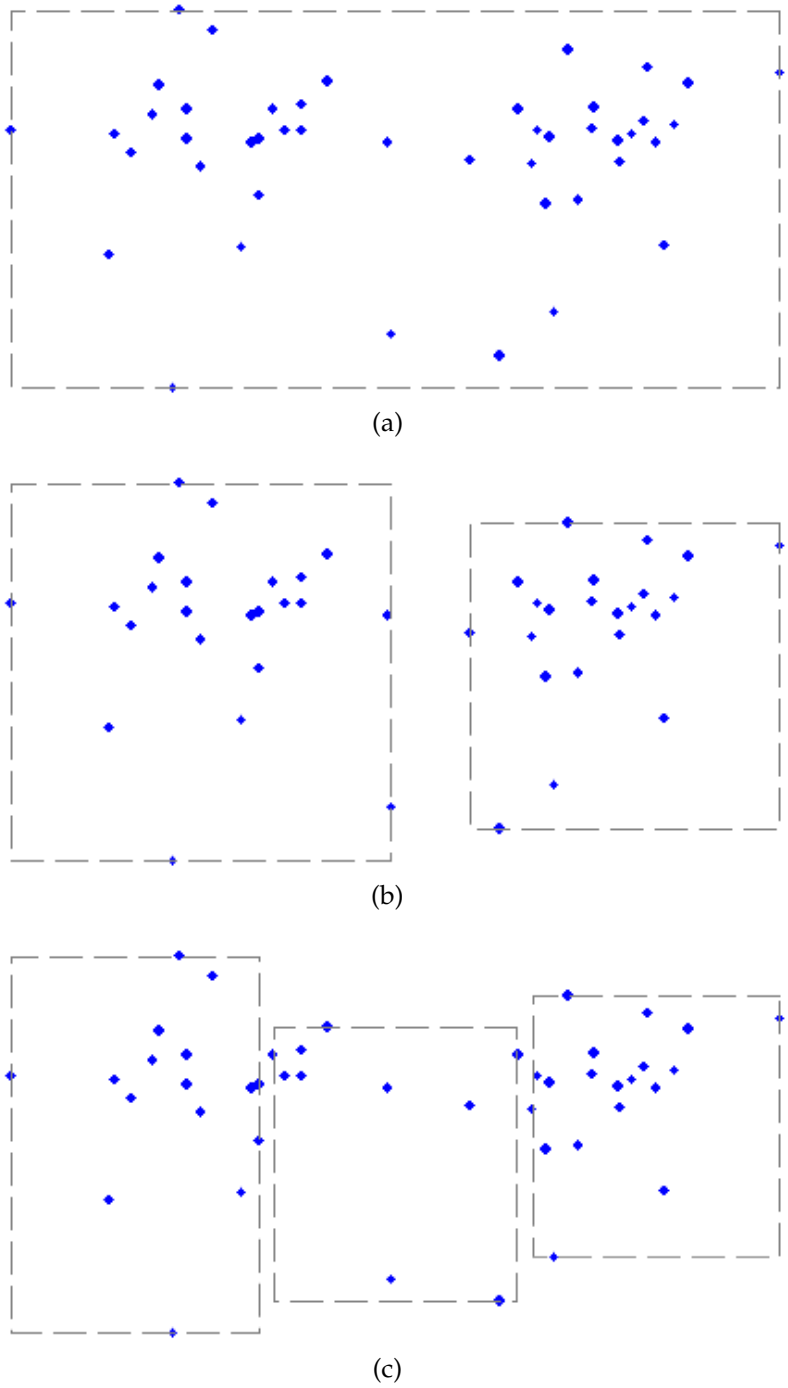


Figure 3.2: Volume preservation on an illustrative data set

Assuming that clusters in the data set have approximately the same size, the cluster radius and the noise distance, respectively, have approximately the radius r of the hyper-sphere with a hypervolume about $V/(c-1)$, when using $c-1$ regular prototypes for the clustering. Since our estimation of the hypervolume is based on a rectangular shape, the radius of a corresponding hyper-sphere would not cover all feature vectors in the hyperbox. Furthermore, huge clusters may be approximated by several prototypes. The feature vectors in border regions of those prototypes should not get high membership degrees to the noise cluster. By all means, different applications require variable definitions regarding outliers. Thus, the noise distance δ can be tuned by a parameter α. Finally we obtain

$$\delta = \alpha r. \tag{3.14}$$

Although, any positive value can be chosen for α, our tests have shown that we achieve good results with $\alpha = 1.5$. In fact, smaller values for α lead to more compact clusters with a higher number of outliers. With $\alpha \to \infty$ NC tends to behave like FCM.

After defining the noise distance we have to specify the minimum membership degree of a feature vector to the noise cluster in order to classify it as an outlier. It is obvious that no constant value will be appropriate to cover all NC partitions. Analogous to the noise distance, also the membership degrees mainly depend on the number of prototypes used for the clustering. In [100] it is already discussed that the probability achieving high membership degrees with FCM, decreases with an increasing number of prototypes. The lower bound for highest membership degrees is of course $1/c$. Of course, the noise distance affects the membership degrees to the regular clusters, too, since a small noise distance forces high membership degrees to the noise cluster and small membership degrees to the regular clusters. So it makes sense to define outliers not only depending on the number of prototypes, but also by the fact which typical high membership degrees occur for a certain partition, which is naturally affected by the noise distance.

As we have discussed above, an outlier may be defined over the expected fraction of noise. With a simple method we can define outliers on the basis of

the feature vector's probability not belonging to a regular cluster. Therefore, we estimate the mean value μ and the standard deviation σ of the membership degrees to the noise cluster and consider a feature vector as an outlier, if its membership degree to the noise cluster deviates more than a certain factor β from the mean value. Thus, the function `isOutlier` returns 1 when, according to this definition, a feature vector is an outlier, otherwise the function returns zero:

$$\texttt{isOutlier}(x_j) \quad = \quad \begin{cases} 1 & \text{if} \quad u_{cj} - \beta \cdot \sigma > \mu, \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

with

$$\mu \quad = \quad \frac{1}{n} \sum_{j=1}^{n} u_{cj} \tag{3.16}$$

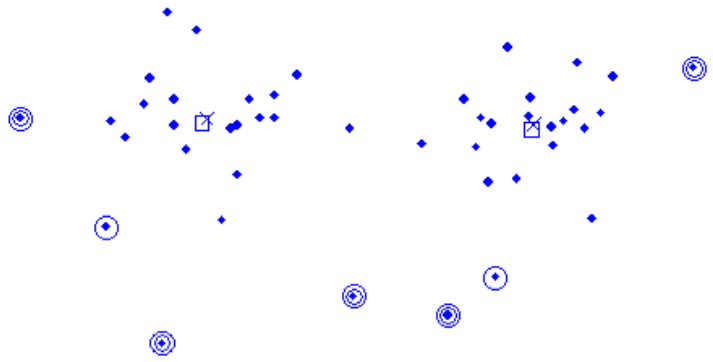
$$\sigma \quad = \quad \sqrt{\frac{1}{n-1} \sum_{j=1}^{n} \left(u_{cj} - \mu\right)^2}. \tag{3.17}$$

Adjusting parameter β one can finally influence the fraction of outliers.

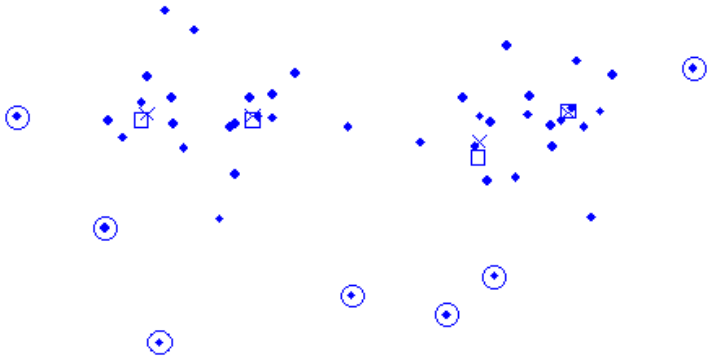
3.5.1 Illustrative Examples

Figure 3.3(a) shows the results of both NC approaches. This data set obviously contains two clusters that are surrounded by some noise points. Using the conventional noise distance for partitioning the data set results in positioning the prototypes as plotted by squares in the figure. Applying the `isOutlier` function with $\beta = 1.4$ the data points marked by the small circle are declared as outliers. The prototypes of the regular clusters are plotted in the figure with the square symbol ($\square$). Since the conventional approach tends to overestimate the noise distance, the optimal cluster centres cannot be found.

When we estimate the noise distance with our volume preserving approach, we obtain a much smaller value for δ. Now, the prototypes that are plotted for this run with the $\times$ symbol are placed closer to the respective cluster centre. In this way, two additional data points were identified as outliers, when



(a) Partitioning with 2 regular prototypes



(b) Partitioning with 4 regular prototypes

Figure 3.3: Noise clustering on an illustrative data set

we use the `isOutlier`-function again with $\beta = 1.4$. The outliers found with the new δ are marked by the bigger circle in the figure.

Figure 3.3(b) shows the results on the same data set using four regular prototypes for the clustering. Real-life data sets usually contain cluster structures that differ from our assumption of hyperspherical clusters. Then the cluster structures must be approximated by several prototypes. A noise clustering technique should be able to deal with such challenges. As the figure shows, the conventional NC cannot find any outlier in this example. This is the case, because the noise distance, when estimated by the conventional approach, will be approximately the same as for the two prototypes. But, as it can be easily verified by visual assessment, the distance from the prototypes to the representing data points is significantly smaller compared to

partitioning with only two prototypes. Thus, when the average distance decreases with an increasing number of prototypes and the noise distance is almost constant, or even worse, increasing, then results similar to the one above will be obtained.

With our volume preserving approach we obtain again the same result as with partitioning with two prototypes. The noise distance is, according to equation (3.14), much smaller. Therefore, the cluster centres can be placed better and distant points will be declared as outliers. The outliers found with the new δ are marked again by bigger circles in the figure. This noise clustering technique can even be applied, when we are not interested in finding a cluster structure, but only want to identify outliers.

3.6 Cluster Validity

Fuzzy clustering algorithms as described above need some parameters to be provided by an expert. A very crucial parameter is the expected number of clusters c to be found in the data set. Admittedly, this is something we often do not know in advance. Also the fuzzifier needs to be adjusted depending on the data set and, of course, depending on the desired fuzziness of the partition. Finally, fuzzy clustering algorithms only find local minima and must be run repeatedly to assure a stable result.

For high-dimensional data sets it is not possible to validate a partition by visual inspection. The objective function itself that is minimized by the optimization scheme is not sufficient to compare partitions with different numbers of prototypes. Assuming an optimal placement of the prototype vectors, an increasing number of prototypes is naturally associated with decreasing intra-cluster distances and thusly smaller J.

Actually, two kinds of validity measures are commonly used to evaluate clustering partitions, namely global validity measures and local validity measures. Compatible cluster merging (CCM) [44, 56] is a commonly used representative for the latter that can be applied on Gustafson-Kessel clustering partitions and related ones for line detection. The objective of CCM is to

estimate the optimal number of clusters during the clustering process. CCM starts with a relatively large number of prototypes and merges clusters that are compatible according to a compatibility relation. CCM is quite sensitive to initialization. Merged clusters need some iterations to stabilize. Parameters need to be carefully adjusted to avoid that all clusters will be merged to one or the other extreme – no clusters will be merged. A robustification of compatible cluster merging can be found in [30]. While only particular applications are suitable for local validity measures, global validity measures are commonly used on a variety of clustering problems. We will briefly discuss some global validity measures in the following. Many other validity measures, defined to measure to what degree the data objects are similar inside one cluster while dissimilar between different clusters, have been described in the literature, see e.g. [9, 21, 22, 44, 101].

Partition Coefficient

The partition coefficient F is a simple validity measure based on the idea that good partitions are characterized through clearly assigned feature vectors [7]. The more crisp the membership degrees the better the partition

$$F = \frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2.$$

Good partitions are rated by values tending to $F \to 1$. Week partitions will be rated about $1/c$. In practice the partition coefficient gives only week hints regarding the appropriate number of clusters. The main problem is the strong dependency of F on the number of prototypes used for the clustering. Of course, an increasing number of prototypes improves the chances to explore the data set profoundly and high membership degrees can be obtained for data near the cluster centres. However, the amount of border regions between clusters increases, too. Prototypes compete there for the data which results in relatively small membership degrees. This is the reason why F often rates partitions with many prototypes badly.

Non Fuzzyness Index

The non fuzzyness index (NFI) is an improvement of the partition coefficient

$$NFI = \frac{cF - 1}{c - 1}.$$

As mentioned above, the partition coefficient strongly depends on the number of prototypes used for the clustering. [100] explains in detail why the expected value of F decreases with an increasing number of prototypes. According to this the probability to obtain high values for F becomes smaller. The domain is between $1/c$ and 1. The comparison of different partitions is thusly only meaningful if the number of prototypes is considered, too. The non fuzzyness index achieves this by linear transformation of F to a range between 0 (week partitioning) and 1 (hard partitioning) [89].

Proportion Exponent

Windham proposed in [100] the proportion exponent validity measure

$$P = -\log_2 \left(\prod_{j=1}^{n} \left(\sum_{k=1}^{\mu_j^{-1}} (-1)^{k+1} \binom{c}{n} (1 - k\mu_j)^{c-1} \right) \right)$$

with $\mu_j = \max_{1 \leq i \leq c} u_{ij}$. The number of feature vectors is included in the calculation of P. Therefore, data sets of different size are not comparable regarding their cluster validity. Already the existence of one single membership degree near 1 initiates the proportion exponent to state a positive evaluation even if the remaining membership degrees are rather fuzzy. Since an increasing number of prototypes naturally increases the probability to obtain at least one high membership degree, P tends to favour partitions that result from clustering with many prototypes.

3.7 Visual Validation of Clustering Results

The measures above have one thing in common – they condense the clustering result to a single value which is associated with a huge loss of information. Moreover, it cannot be derived which part of the data needs to be

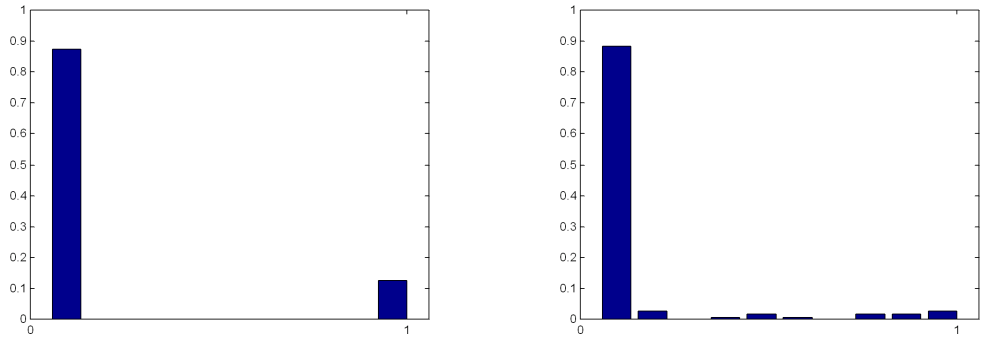


Figure 3.4: Distribution of membership degrees for two out of nine prototypes clustering the Cube data set

investigated more. Many more measures have been developed so far which cannot be considered here, see e.g. [9, 31, 33, 44, 101]. In the following we discuss the visual validation of clustering results.

Visualizing Frequencies of Membership Degrees

An intuitive approach validating fuzzy clustering results is to visualize the distribution of the membership degrees. Denominate the crisp result as the ideal case, i.e. each datum is assigned to exactly one cluster with membership degree 1 and to all other clusters with membership degree 0, one would expect that the relative frequency of the membership degree 1 should be $1/c$ and accordingly the relative frequency of the membership degree 0 should be $(c-1)/c$. The chart diagram for the ideal case of crisp membership degrees shows a distribution of membership degrees as follows: a value of $(c-1)/c$ on the left side and $1/c$ on the right side and zero values in between.

Figure 3.4 shows the distribution for two kinds of prototypes that will be obtained when clustering the Cube data set with nine prototypes. Membership degrees are grouped into ten classes obtained by subdividing the possible membership range into ten equisized sections. Seven clusters will be represented by a prototype like the one that is depicted in figure 3.4(a). A suchlike prototype is ideal. The figure reveals that the prototype repre-

sents some data clearly. To the rest of the data this prototype has only very low membership degrees. The distribution of membership degrees for the second prototype, depicted in figure 3.4(b), state a non-ideal cluster. Two prototypes of this kind are obtained for this example. It is not surprising that these two prototypes represent parts of one identical data cluster of the Cube data set. The relatively large amount of medium membership degrees indicates that a data cluster was improperly represented by more prototypes than appropriate.

In [49] an improvement to visualize membership distributions is proposed. A scaling is carried out in such way that in the ideal case (crisp partitioning) the chart diagram would show a value of 1 on both, the left and the right side. For this purpose a weighting factor is introduced when counting the frequencies of the membership degrees.

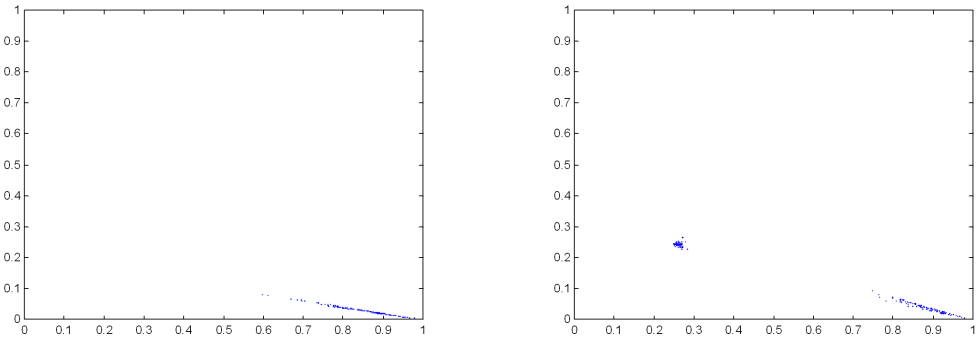
Visualizing Membership Degrees of the two most Competing Clusters

Another approach visualizing clustering results is to consider for each feature vector x_j the cluster with the highest membership degree, say i, and the cluster yielding the second highest membership degree, say ℓ. Then for each feature vector x_j a point is plotted at the coordinates $(u_{ij}, u_{\ell j})$.

Consequently, all points must lie within the triangle defined by the points $(0,0)$, $(0.5,0.5)$ and $(0,1)$, since the first coordinate must always be larger than the second one and according to the probabilistic constraint we have $u_{ij} + u_{\ell j} \leq 1$.

Regarding the ideal case all points would be plotted near the point $(1,0)$. Ambiguous data that are shared by two clusters are indicated by points near $(0.5,0.5)$. Points near $(0,0)$ originate from data that cannot be assigned to any cluster clearly. Such points either indicate that an improper number of prototypes was chosen for the clustering or that the clustering technique is not appropriate for the data structure.

Figure 3.5 shows the results of this technique on two different partitionings of the Cube data set. The left figure shows an almost ideal clustering of the Cube data set. Eight prototypes where used here which complies with the



(a) Clustering with 8 prototypes (b) Clustering with 5 prototypes

Figure 3.5: Visualizing membership degrees of the two most competing clusters of the Cube data set

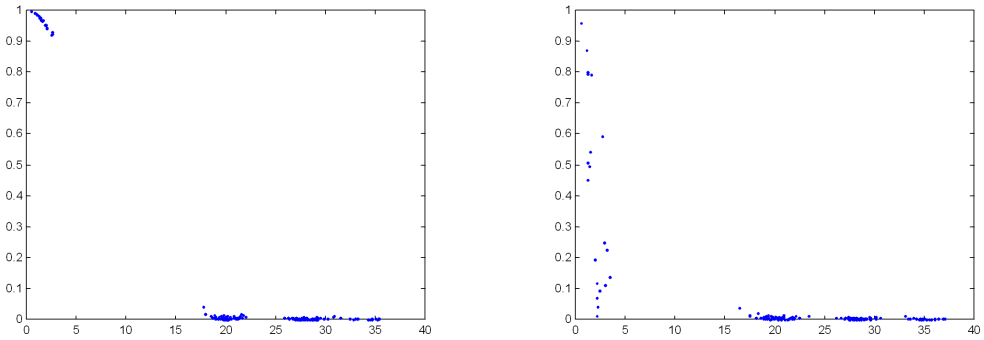
data set's natural number of clusters. Most of the points are concentrated near $(1, 0)$. Of course, memberships are decreasing smoothly when using a moderate fuzzifier $m = 2$ as we did here. Therefore, some points can be found slightly elongated tending to $(0.5, 0.5)$.

The right figure clearly shows that some data is badly represented by the available prototypes. This figure shows the clustering result using only five prototypes. Comparable to the left figure, some points lie near $(1, 0)$. These points are well approximated by some prototypes. Contrary, the presence of a considerable amount of points lying between $(0.5, 0.5)$ and $(0, 0)$ indicates a misjudged number of prototypes used for the clustering.

Visualizing Membership Degrees over Intra-Cluster Distances

The above approaches solely consider the membership degrees for the visualization. Even more insight can be gained from a plot of the membership degrees over the distances for each cluster, as stated in [49]. For each cluster i a point is plotted for each datum x_j at (d_{ij}, u_{ij}).

For an ideal graph one would expect high membership degrees for small distances and low membership degrees for large distances. When cluster centres are not chosen appropriately two different effects can be observed. If the number of prototypes is chosen too small, one cluster has to cover two or



(a) An ideal cluster (b) An unfavourable cluster

Figure 3.6: Visualizing membership degrees over intra-cluster distances of the Cube data set (clustering with nine prototypes)

more data clusters. In this case, fewer points are plotted in the upper left part of the graph. We will rather find more points in the middle of the graph. If the number of prototypes is chosen to high, then two or more clusters compete and share the same data cluster. Consequently, even for small distances low membership degrees occur which will be reflected by points in the lower left part of the corresponding diagram.

Figure 3.6 shows the results of this technique on two interesting clusters resulting from clustering the Cube data set using nine prototypes. The left graph shows an ideal cluster. Data with small distances to the cluster centre yield high membership degrees. The gap to the rest of the data emphasizes the compactness of the depicted cluster. The aforesaid clustering yields seven clusters of that kind. The continuous slide from high to low membership degrees on the right graph indicates that this cluster is not well separated from another cluster. Indeed, two prototypes represent the remaining data cluster.

Figures 3.7(a) and 3.7(c) show two clusters of a clustering with five prototypes. Figures 3.7(b) and 3.7(d) show the data set and all prototypes obtained by the clustering. The prototype that is visualized by the respective left figure is marked by a surrounding circle. An interesting fact in figure 3.7(a) is that some data points obtain different membership degrees even though

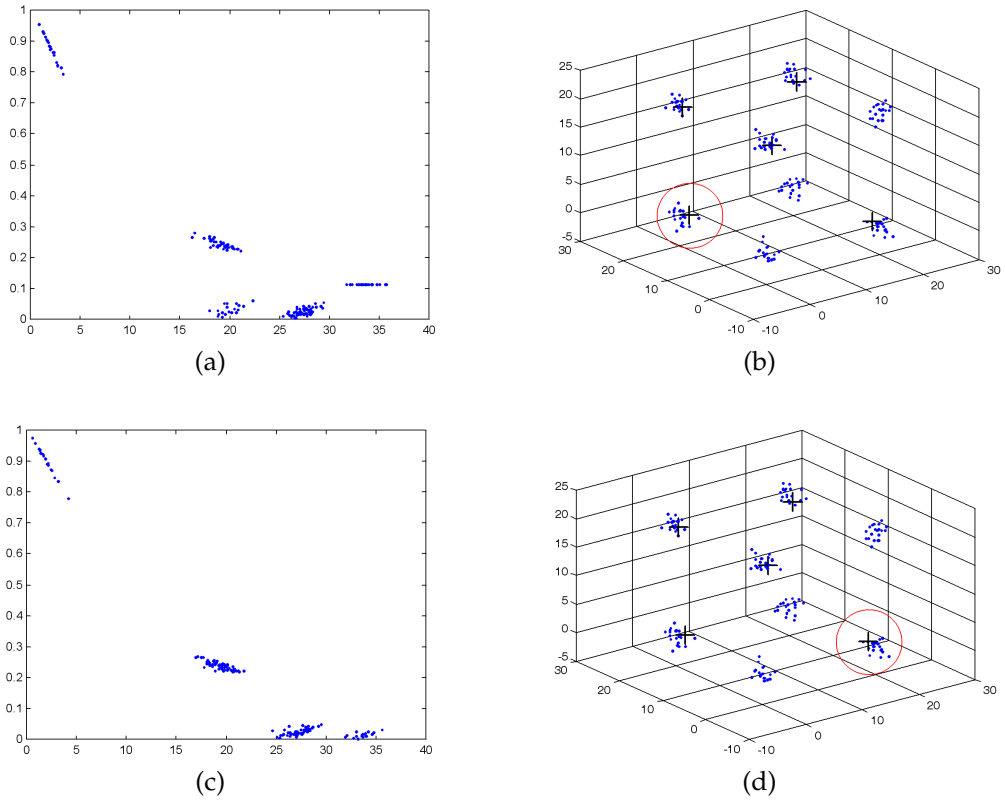


Figure 3.7: Visualizing membership degrees over intra-cluster distances of the Cube data set (clustering with five prototypes)

their distance to the prototype is similar. The reason for this can be seen in figure 3.7(b). There are two data clusters that are not well represented by any prototype vector. Another data cluster which has approximately the same distance is well represented. Thus, the prototype obtains only small membership degrees to the data points in this cluster.

Figure 3.7(c) shows a prototype where this effect cannot be observed. Data, which has similar distances to this prototype, obtain similar membership degrees, too. This is evident as figure 3.7(d) shows. All other prototypes have approximately the same distances to the data in question.

Visualization of Clustering Results by Modified Sammon's Mapping

Earlier we discussed Sammon's mapping as a common representative for multidimensional scaling. It is also common knowledge that Sammon's mapping is not applicable to larger data sets since its time complexity is about $O(t \cdot n^2)$, where t is the algorithm's unknown number of iterations. In [29] an efficient modification is described that aims at visualizing fuzzy clustering results. This approach was finally refined and described in [55]. The algorithm maps the cluster centres and the data in such a way that the original distances between the clusters and the feature vectors will be preserved as good as possible. This means that the algorithm only considers distances of a usually small number c of prototype vectors to the n feature vectors minimizing the following functional

$$E = \sum_{i=1}^{c} \sum_{j=1}^{n} (u_{ij})^m (d_{ij}^x - d_{ij}^y)^2 \tag{3.18}$$

where d_{ij}^x represents the distance between feature vector x_j and the cluster v_i measured in the original p-dimensional space, while d_{ij}^y represents the distance between the projected cluster centre z_i and the projected data y_j in the low-dimensional space (usually the plane). The proposed algorithm uses the gradient descent method taking the partial derivatives of E yielding

$$\frac{\partial E}{\partial y_k} = -2 \sum_{i=1}^{c} u_{ik}^m (d_{ik}^x - d_{ik}^y) \frac{y_k - z_i}{d_{ik}^y}.$$

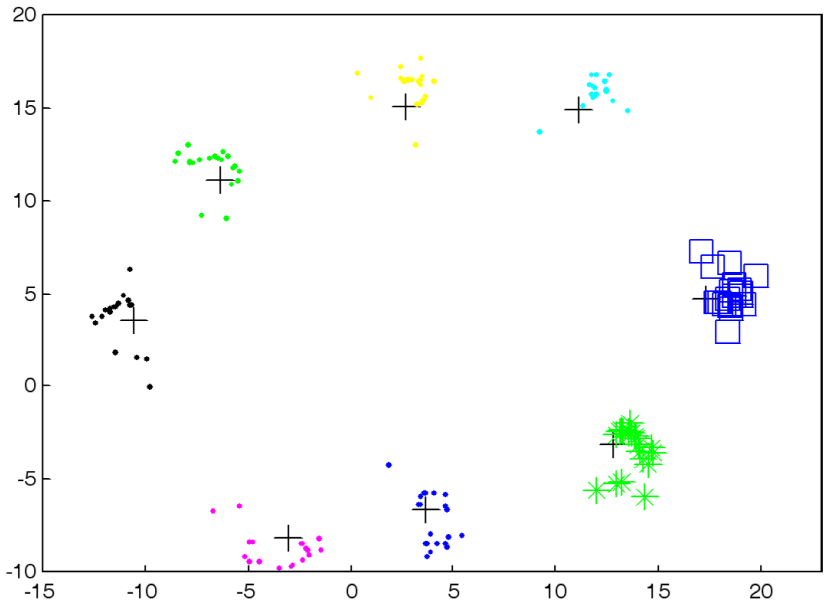
For each iteration t the cluster centres z_i in the plane will be determined according to

$$z_i^{(t)} = \frac{\sum_{j=1}^{n} (u_{ij})^m y_j^{(t)}}{\sum_{j=1}^{n} (u_{ij})^m}$$

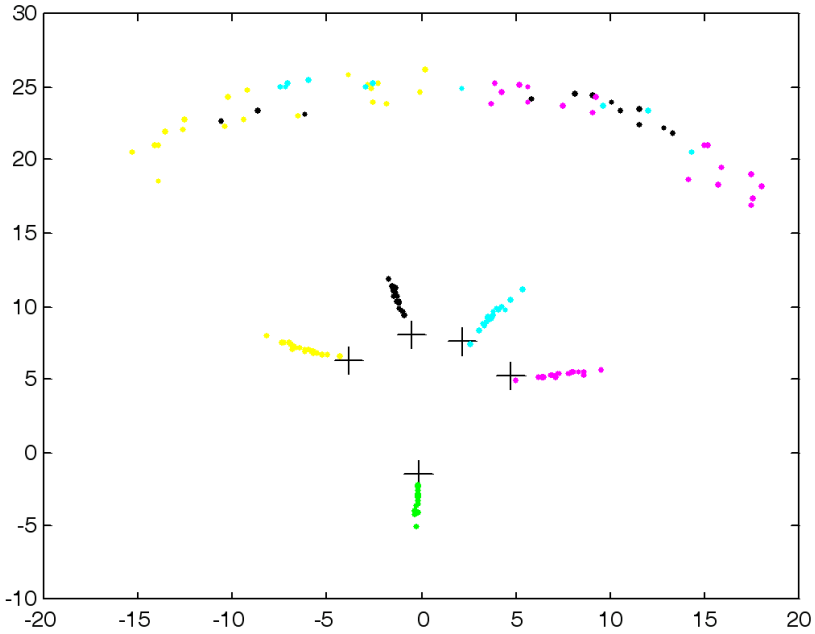
and the mapping of each projected feature vector y_k according to

$$y_k^{(t-1)} = y_k^{(t-1)} - \Delta y_k^{(t)}$$

where $\Delta y_k^{(t)} = \partial E^{(t)} / \partial y_k^{(t)}$. The algorithm finally converges when the error terms of two consecutive iterations are nearly identical, i.e. $\|E^{(t-1)} - E^{(t)}\|$ becomes smaller than a user defined threshold ε, or a maximum number of iterations t_{max} has been reached.



(a) Clustering with 8 prototypes



(b) Clustering with 5 prototypes

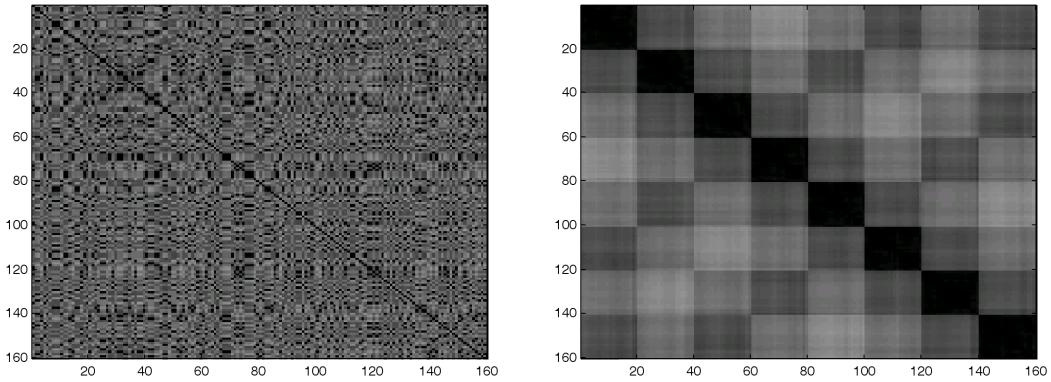
Figure 3.8: Visualization of partitions of the Cube data set using the modified Sammon's mapping

Even though the complexity per iteration is quite low, many iterations will be needed until convergence as our tests reveal. Figure 3.8 shows two mappings of the Cube data set obtained using the described method. The upper graph shows a mapping using eight prototypes for the clustering and the mapping, respectively. The mapping reflects the original cluster structure approximately presenting eight well separated clusters and their corresponding prototypes. The lower graph shows the mapping of a five-cluster-partition of the same data set. It is clearly visible that some data are in the proximity of the cluster centres and thusly well represented. However, there are also some data which are not covered by any cluster clearly which meets our expectations due to the unsuitable choice of prototypes.

Visual Assessment of Cluster Tendency

VAT, Visual Assessment of Cluster Tendency, is a tool to visualize pairwise dissimilarity information of objects $X = \{x_1, \ldots, x_n\}$ as a square image with n^2 pixels [8]. VAT reorders the data objects so that the image highlights potential cluster structures. The reordering algorithm in VAT is similar to finding a minimal spanning tree, but with two differences. One difference is that VAT does not generate the minimal spanning tree but it finds the order in which the vertices are added to the tree. The second difference is that VAT requires the definition of an initial vertex. Reordering begins by building a dissimilarity matrix R with $R_{ij} = |x_i - x_j|$ and taking the object that has the largest distance to any other object and finding the object closest to it. VAT then finds the object closest to either of the first two objects. This procedure is repeated until all objects have been considered in the reordering.

Figure 3.9 shows two dissimilarity images of the Cube data set using the normalized R_{ij}-value as grey tone for each feature vector. Whereas dark pixels correspond to nearby objects do light pixels stand for distant objects. The left graph shows the unordered Cube data set. The right graph shows the dissimilarity image produced by VAT. The eight data clusters of the Cube data set, visualized by dark blocks, can be easily found in the graph. Their separation to other clusters is indicated by the sharp contrast to neighbour-



(a) Dissimilarity image of the unordered Cube data set

(b) Dissimilarity image of the Cube data set produced by VAT

Figure 3.9: Visual assessment of cluster tendency of the Cube data set

ing blocks. The clusters' sizes can also be read from the blocks' dimensions.

As a modification of VAT, bigVAT allows the visualization for larger data sets reducing computational complexity by performing a quasi-ordering of the objects and enableling graphical representations larger than usual screen sizes [45]. VCV, Visual Cluster Validity, is related to VAT, but takes the inter-datum distances into account that come from partitioning the data set [37].

3.8 Visualizing Single Clusters

We proposed in [86, 87] an approach – called Single Cluster Visualization (SCV) – to visualize single clusters by projection of the data points onto the plane under the constraint that the membership degrees to clusters are preserved. Note, membership degrees can be obtained directly when using a fuzzy clustering algorithm (e.g. fuzzy c-means), but also when calculating membership degrees after the partitioning, which can be done for any prototype-based clustering algorithm. To achieve the objective of membership preservation, we adopt the noise distance aspect of the noise clustering technique [17].

Noise clustering is based on the introduction of an additional noise cluster that is supposed to contain all feature vectors that are about a certain

distance, the noise distance δ, away from all other prototype vectors. This means that the prototype v_c for the noise cluster c has no parameters. The clustering scheme differs only in one point from k-means or fuzzy c-means. When calculating the membership degrees the distance of the feature vector x_j to the noise cluster v_c is the fixed constant value $d_{cj} = \delta^2$. The proper specification of δ is discussed in [17, 81].

With the objective to place the cluster in the plane, we need two coordinates for each data point. Note that the constraint for the projection is not to preserve the distances d_{ij} but the membership degrees u_{ij}. The idea for our visualization is to compute the distances to the cluster prototypes by means of the membership degrees. To achieve this we consider the usual computation of membership degrees as mentioned in equation (3.5). This provides a very simple connection between membership degrees and distances

$$\frac{u_{ij}}{u_{\ell j}} = \frac{\frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{1}{m-1}}}}{\frac{1}{\sum_{k=1}^{c}\left(\frac{d_{\ell j}}{d_{kj}}\right)^{\frac{1}{m-1}}}} = \left(\frac{d_{\ell j}}{d_{ij}}\right)^{\frac{1}{m-1}}. \tag{3.19}$$

For the purpose of visualization we propose to place the cluster i to be visualized at $(0,0)$ and to choose a second cluster ℓ at $(1,0)$. The cluster at $(1,0)$ is a virtual cluster that contains all feature vectors with the highest membership degree apart from u_{ij}. The intention of this cluster is to collect all feature vectors that are assigned to another cluster than the one we want to visualize. Let us denote the membership degree to the most competing cluster by $u_{\ell j}$. Furthermore, we introduce a noise cluster to cover the clusters apart from i and ℓ. According to the distance of the two chosen cluster prototypes at $(0,0)$ and $(1,0)$, we define the noise distance $\delta = 1$. This means we have $u_{noisej} = 1 - u_{ij} - u_{\ell j}$. According to equation (3.19) this leads to

$$\frac{u_{ij}}{u_{noisej}} = \left(\frac{1}{\hat{d}_{ij}}\right)^{\frac{1}{m-1}}. \tag{3.20}$$

We denote the distance between cluster i and x_j on the plane by $\hat{d}_{ij}$ to emphasize the fact that we do not deal with original distances any more but with

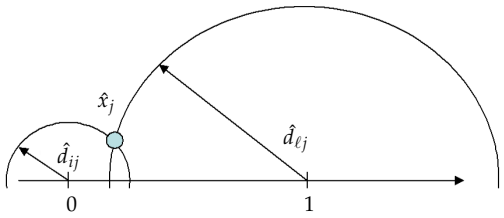


Figure 3.10: Placement of $\hat{x}_j$ in the plane

representative distances with respect to the according membership degrees. Solving equation (3.20) for $\hat{d}_{ij}$ we obtain

$$\hat{d}_{ij} = \left(\frac{u_{noisej}}{u_{ij}} \right)^{m-1} . \tag{3.21}$$

Analogously, we obtain for the second cluster ℓ

$$\hat{d}_{\ell j} = \left(\frac{u_{noisej}}{u_{\ell j}} \right)^{m-1} . \tag{3.22}$$

This approach enables us to visualize some useful aspects:

- which feature vectors can be assigned clearly to the cluster i of interest,
- if a feature vector cannot be assigned to i, is there another cluster ℓ, where the vector can be assigned to,
- which feature vectors are near to one or more prototypes apart from i and ℓ,
- are there feature vectors that cannot be assigned to any cluster clearly.

With equation (3.21) one can compute the distance of each feature vector x_j to the cluster i, so that it is possible to draw a circle around $(0, 0)$ as one hint for the feature vector's position in the plane. With the distance to the other cluster $(1, 0)$, one could draw another circle around the cluster centre. The intersection point of these two circles would be the position of the new feature vector in the plane. Note that there are usually two intersection points. We only consider the one above the x-axis. It is also possible that the two

circles do not intersect at all. This case will be discussed in the section on implementation aspects.

Figure 3.10 illustrates this approach. The small circle that represents the potential coordinates of $\hat{x}_j$, can be drawn with radius $\hat{d}_{ij}$ obtained from equation (3.21). Analogously, the bigger circle can be drawn with radius $\hat{d}_{\ell j}$ that we get with equation (3.22). The intersection point of these two circles represents the feature vector $\hat{x}_j$ in the plane.

3.8.1 Implementation Aspects

Apart from the clustering itself, which leads to the membership degrees u_{ij} another parameter affects the transformation, namely m (see equations (3.20, 3.21, 3.22)). A priori, one would take the same value for m as for the clustering. But it can also be useful to modify this parameter. Practical tests have shown that in some cases, e.g. when a feature vector is very close to a prototype vector, no intersection point can be obtained in the plane and consequently the membership degrees to the respecting feature vector cannot be preserved exactly.

The rules shown in algorithm 7 handle such cases while trying to preserve membership degrees approximately. Let us denote the transformed data set $\hat{X}$. The two circles have no intersection point only in the case, when the feature vector is very close to one of the clusters. In this case, we place the point on the x-axis, i.e. $\hat{x}_{2j} = 0$. The rest of the rule tries to find the proper position for $\hat{x}_j$ on the x-axis balancing the distances to cluster $(0,0)$ and cluster $(1,0)$. If the distance to both clusters is relatively small, say $\max(\hat{d}_{ij}, \hat{d}_{\ell j}) < 1$, then we compute a position between both clusters in relation to $\hat{d}_{ij}$ and $\hat{d}_{\ell j}$. Otherwise, which means one or both clusters are about a distance of 1 or further away from the feature vector, we distinguish whether cluster $(0,0)$ or cluster $(1,0)$ is nearer. If the distance of x_j to cluster $(0,0)$ is higher than the distance to cluster $(1,0)$ then $\hat{x}_j$ will be placed to the right of cluster $(1,0)$ at $\hat{x}_j = (1 + \hat{d}_{\ell j}, 0)$. If the distance $\hat{d}_{ij}$ to cluster $(0,0)$ is smaller than the distance $\hat{d}_{\ell j}$ to cluster $(1,0)$ then $\hat{x}_j$ will be placed to the left of cluster $(0,0)$ at $\hat{x}_j = (-\hat{d}_{ij}, 0)$. This concept enables an accurate placement of data points

Algorithm 7 Placement of $\hat{x}_j$ on the x-axis

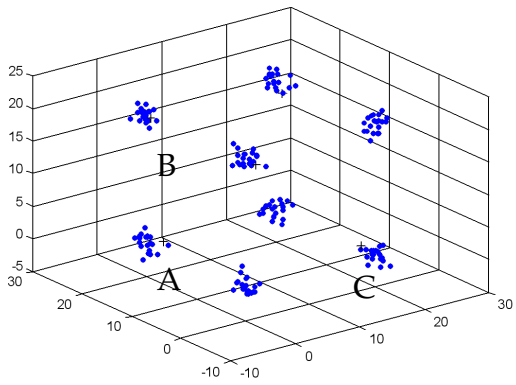
```
if no intersection point then
  x̂_{2j} = 0
  if max(d̂_{ij}, d̂_{ℓj}) < 1 then
    x̂_{1j} = d̂_{ij} / (d̂_{ij} + d̂_{ℓj})
  else
    if d̂_{ij} > d̂_{ℓj} then
      x̂_{1j} = 1 + d̂_{ℓj}
    else
      x̂_{1j} = −d̂_{ij}
    end if
  end if
end if
```

relative to the nearest cluster at least. However, it is not essential to know the exact distance of the feature vector to the other cluster, since the distance is quite large in fact.

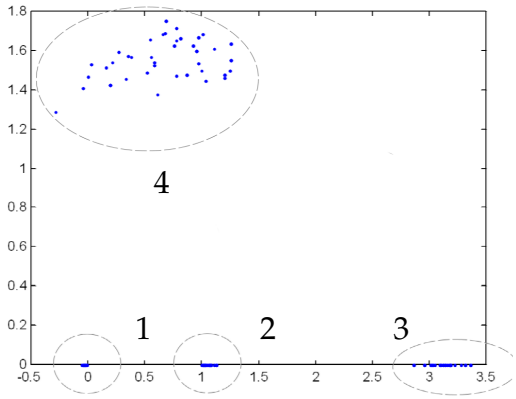
With these rules the membership degrees cannot be preserved exactly, but approximated intuitively. Alternatively, one can avoid this kind of approximation by modifying parameter m for the transformation process. Small values $m \to 1$ prevent that no intersection point can be met. Otherwise, one can set higher values for m to force placements on the x-axis. Such transformations may not be that differentiated, but information can be reduced to some essential facts if needed. Generally, data points situated left from 0.5 on the x-axis can be assigned to cluster $(0, 0)$, while data points on the other side belong to another cluster.

3.8.2 Illustrative Examples

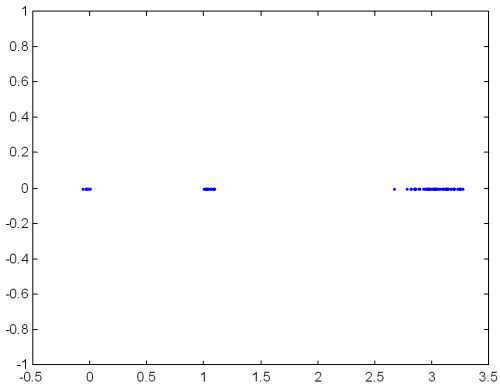
Let us first apply our visualization method to an artificial data set. The Cube data set (see figure 3.11(a)), that we have used before already, consists of eight well separated clusters, which are in the corners of an imaginary 3-dimensional cube. A fuzzy c-means partition of the data set with five prototypes is shown in the figure. Of course, eight prototypes would be the best



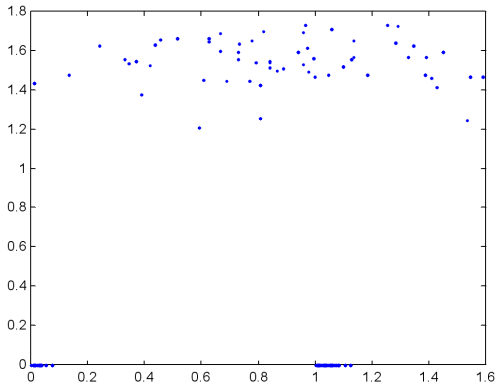
(a) Clustering of the Cube data set with 5 prototypes



(b) Transformation of the Cube data set from the perspective of prototype *A*



(c) Transformation of the Cube data set from the perspective of prototype *B*

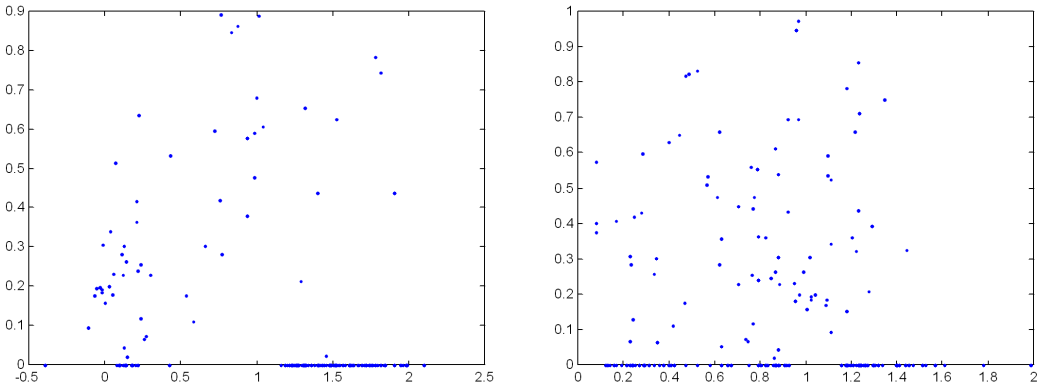


(d) Transformation of the Cube data set from the perspective of prototype *C*

Figure 3.11: SCV - an illustrative example

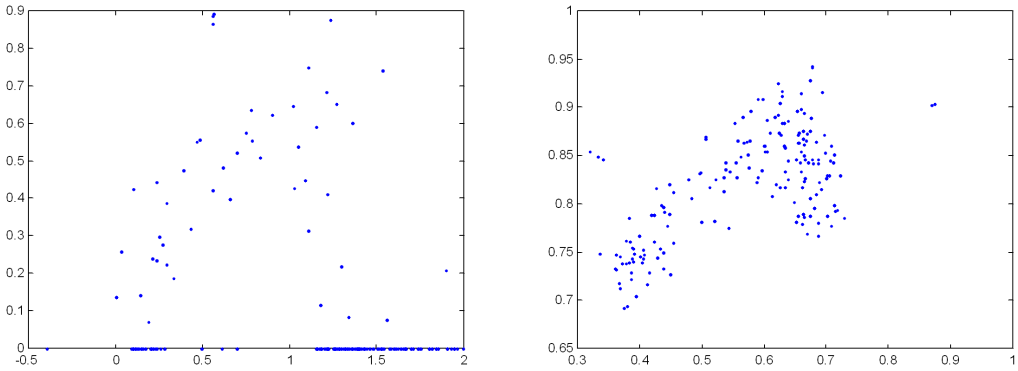
choice to partition the Cube data set. Thus, we can illustrate with this partition which information one can get from the visualization tool, if the number of clusters is chosen wrongly.

Figure 3.11(b) shows the SCV-transformation of the Cube data set from the perspective of prototype *A*. Clearly four groups of data points can be observed (circled with a dashed line). The data points in group 1 are those, which can be clearly assigned to prototype *A*. Data points that are located in group 2 are those, which are not assigned to prototype *A* at all, but to another prototype. Note, a partition that only consists of these both groups is ideal.



(a) (b)

Figure 3.12: SCV transformations of single clusters of the Wine data set



(a) Transformation with $m = 1.2$ (b) Transformation with $m = 1.02$

Figure 3.13: The effect of parameter m on SCV transformations

Group 3 stands for feature vectors, which are not assigned to prototype A and not to any other prototype. Instead, the data points have approximately the same membership degree to two or more prototype vectors (but not to prototype A). Group 4 represents feature vectors that have approximately the same membership degree to prototype A and another prototype.

Figure 3.11(c) shows the transformation of the Cube data set from the perspective of prototype B. At first sight one can notice that group 4 is absent. That means in fact that no other prototype than prototype B has high membership degrees to the data points in group 1. A closer look reveals that the

distance of prototype B to some misrepresented data is higher comparing to other prototypes, such as prototype A and C. All other data points that could contribute to group 4 are clearly represented by some prototypes.

The transformation of the Cube data set from the perspective of prototype C is shown in figure 3.11(d). Now group 3 is missing in the plot. This becomes evident, because all data points that are underrepresented are directly between prototype C and at least one other prototype. As we have discussed above, group 3 only occurs when data points have low membership degrees to the regarding prototype and approximately equal membership degrees to two or more other prototypes. Since prototype C is at least as near to the data as other prototypes, group 3 cannot be formed.

Figure 3.12 shows some results on the well known Wine data set. The figure shows as an example two clusters of a partitioning with four prototypes. The left one is a visualization of a quite compact cluster. Data points left from 0.5 on the x-axis whose component on the y-axis is greater than zero have only small membership degrees to the cluster $(1, 0)$ even if their distance to cluster $(0, 0)$ seems to be far. This is due to the relatively small fuzzifier that is used during the clustering. The cluster shown in figure 3.12(b) is much more overlapping other clusters as the points on the x-axis, fairly in the middle between both clusters, indicate. As mentioned above, using small values for m leads to rather sensitive transformations. Even a relatively small membership degree to a certain cluster attracts the data points in the transformation. To smooth this effect it is advisable to decrease m for the transformation or increase m for the clustering if possible.

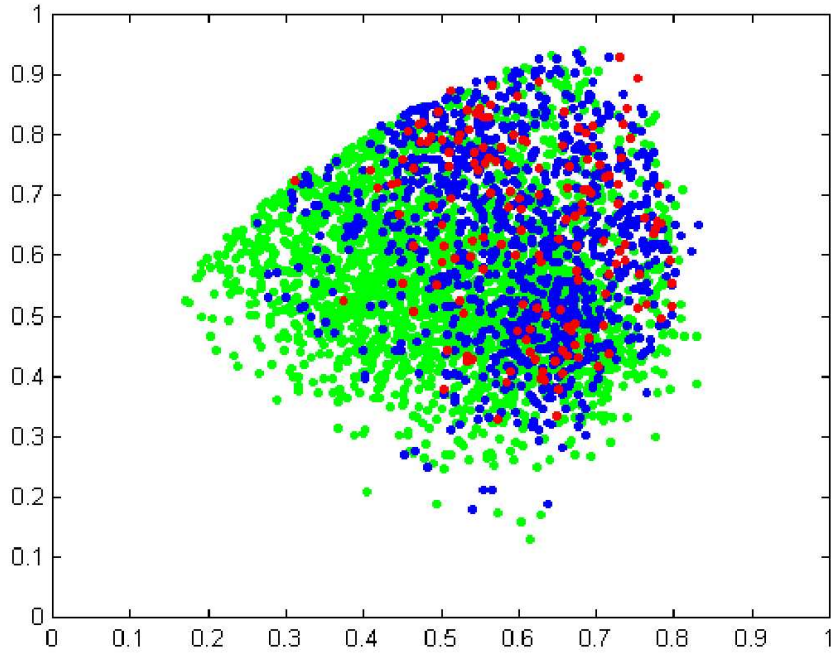
The effect of decreasing m for the transformation is shown in figure 3.13. While figure 3.13(a) shows the transformation of a cluster of the Wine data set with $m = 1.2$, figure 3.13(b) shows the same cluster transformed with $m = 1.02$. The changeover from cluster $(0, 0)$ to cluster $(1, 0)$, which is the imaginary line at 0.5 through the x-axis, is rather sparse. This fact indicates a compact cluster with only few feature vectors which cannot be assigned clearly to any cluster.

3.8.3 Discovering Weather Clusters Impacting Air Traffic Management

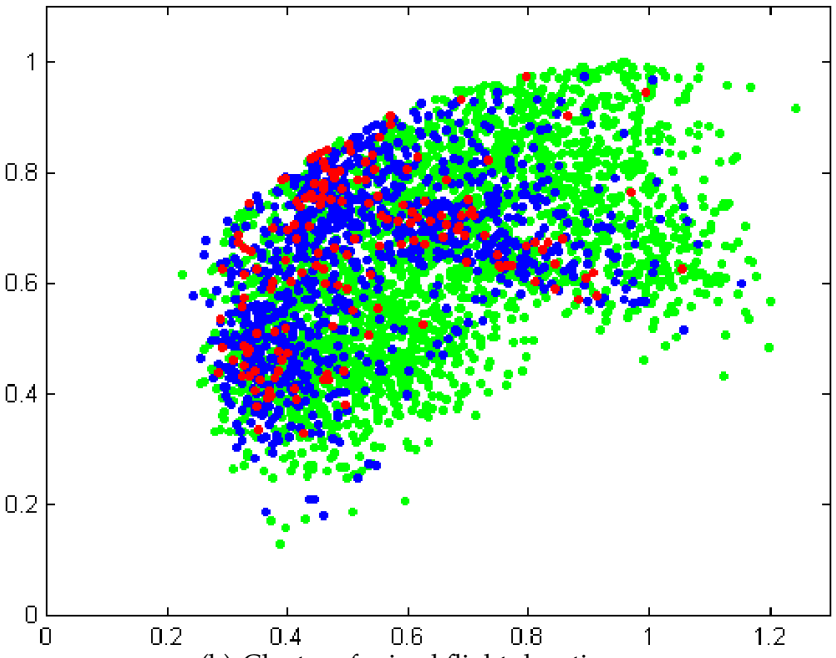
In this section we discuss the results of SCV applying it to the weather data set [84]. Three categories are defined (short, medium and long flights) according to the flight durations in the TMA. Short flights correspond to flights taking up to one minute more than the average flight duration in the TMA. Medium flights exceed the average flight duration up to eight minutes. The remaining flights correspond to the longer flights.

Figure 3.14(a) shows one cluster of a three-cluster-partition. Short flights are visualized by green points, medium flights by blue points and long flights by red points. All feature vectors left from 0.5 on the x-axis have their highest membership degree to the cluster we try to visualize here. At first sight it is visible that no compact cluster could be found. The changeover from cluster i to cluster ℓ is quite fluent. According to the visualization no clear border between cluster i and another cluster can be drawn. A second cluster, depicted in figure 3.14(b), represents flights of all three categories. Estimating flight durations based on the cluster's average flight duration produces in comparable cases a considerable variance and poor predictions accordingly. Borders between cluster i and cluster ℓ cannot be decided.

These visualizations reveal that flight durations can be partly classified using weather data as figure 3.14(a) evinces. However, the entire data set should not be analyzed only by partitioning methods. Some regions in the feature space seem to be more complicated and flight duration categories cannot be separated linearly. Recent studies applying support vector machines could improve prediction quality and underline our assumptions [61, 98].



(a) Cluster of short flight durations



(b) Cluster of mixed flight durations

Figure 3.14: Visualization of weather clusters with SCV

3.9 Prototype-Based Outlier Detection

The detection of outliers as we proposed in [51] is a modified version of the one proposed in [92] and is composed of two different techniques namely clustering and statistical outlier detection.

The technique is described by means of an alternating procedure that repeats clustering of the data set as a first step and removing outliers as a second step. In the clustering step the data set will be partitioned with a prototype-based clustering technique, such as *k*-means or fuzzy *c*-means, so that the feature space is approximated with an adequate number of prototypes. According to the clustering algorithm, the prototypes will be placed in the centre of regions with a high density of feature vectors. Even though outliers are far away from the typical data they influence the placing of the prototypes [25].

Algorithm 8 describes the outlier detection procedure schematically. After partitioning the data, feature vectors are considered as belonging only to a single cluster. Note, when using a fuzzy clustering algorithm a defuzzification has to be done. For each attribute t of the feature vectors of the considered cluster, the mean value, i.e. v_i, and the standard deviation σ_{v_i} has to be calculated. For the vector x_j with the largest distance to the mean vector, which is assumed to be an outlier, the value $z_j^{(t)}$ of the z-transformation

$$z_j^{(t)} = \frac{\left| x_j^{(t)} - v_i^{(t)} \right|}{\sigma_{v_i^{(t)}}}$$

for each of its components is compared to a critical value which depends on the sample size. If one of these values is higher than the respective critical value, then this vector is declared as an outlier. One can use the Mahalanobis distance as in [92], however, since simple clustering techniques like the (fuzzy) *c*-means algorithm tend to spherical clusters, we apply a modified version of Grubbs' test [35], not assuming correlated attributes within a cluster.

The critical value is a parameter that must be set for each attribute de-

Algorithm 8 Prototype-based outlier detection

Given the data set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$
Clustering of the data set yielding c prototypes v_i
Define critical $z^{(t)}$ for each attribute t
repeat
  for $i = 1$ to c **do**
    for all data x_j in cluster i **do**
      for each attribute t of x_j **do**
        Compute $z_j^{(t)} = \frac{\left|x_j^{(t)} - v_i^{(t)}\right|}{\sigma_{v_i^{(t)}}}$
        if $z_j^{(t)} > z^{(t)}$ **then**
          remove x_j
        end if
      end for
    end for
    Compute new positions for all prototypes v_i
    Update critical $z^{(t)}$ for each attribute t according to the new cluster size
  end for
until no outlier found

pending on the specific definition of an outlier. One typical criterion can be the maximum number of outliers with respect to the amount of data [48]. Eventually, large critical values lead to smaller numbers of outliers and small critical values lead to very compact clusters. Note that the critical value is set for each attribute separately. This leads to an axes-parallel view of the data, which in case of axes-parallel clusters leads to a better outlier detection than the (hyper)-spherical view on the data.

If an outlier was found, the feature vector has to be removed from the data set. With the new data set, the mean value and the standard deviation have to be calculated again for each attribute. With the vector that has the

largest distance to the new centre vector, the outlier test will be repeated by checking the critical values. This procedure will be repeated until no outlier will be found anymore. The other clusters are treated in the same way.

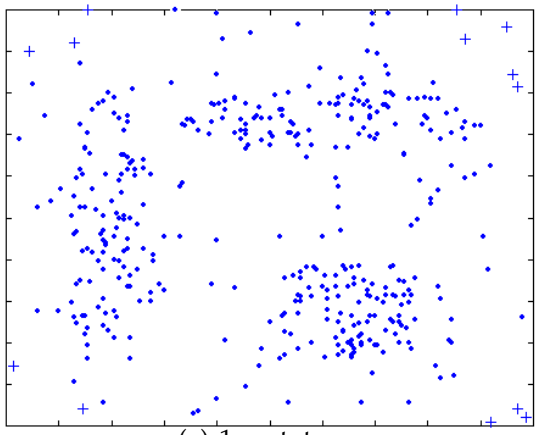
3.9.1 Illustrative Examples

Figure 3.15 shows the results of the proposed algorithm on an illustrative example. The crosses in this figure are feature vectors, which are recognized as outliers. As expected, only few points are declared as outliers, when approximating the feature space with one prototype only (see figure 3.15(a)). The prototype will be placed in the centre of all feature vectors. Hence, only points on the edges are recognized as outliers. Comparing the solutions with three and ten prototypes one can determine that both solutions are almost identical. Even in the border regions, where two prototypes compete for some data points, the algorithm rarely identifies these points as outliers, in accordance to our intuition.

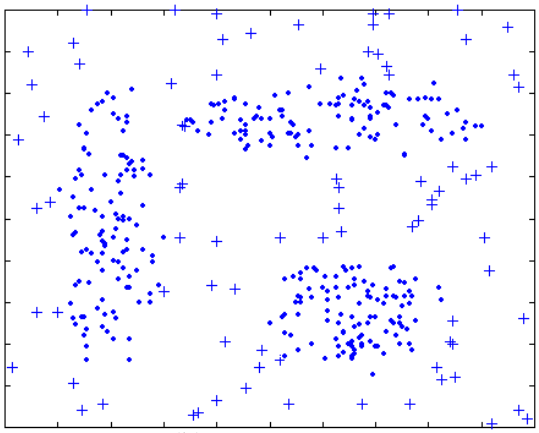
The figure shows that the algorithm can identify outliers in the illustrative data set in a stable way. With only few parameters the solution can be adapted to different requirements concerning the specific definition of an outlier. With the choice of the number of prototypes, it is possible to influence the result in that way that with a larger number of prototypes even smaller data groups can be found. To avoid an overfitting to the data it makes sense in certain cases, to eliminate very small clusters before applying the outlier elimination procedure. However, finding out the proper number of prototypes should be of interest of further investigations.

In case of using a fuzzy clustering algorithm like FCM to partition the data, it is possible to assign a feature vector to different prototype vectors. In that way one can consolidate that a certain feature vector is an outlier or not, if the algorithm decides for each single cluster, or at least for the clusters that yield the highest membership degree, that the corresponding feature vector is an outlier.

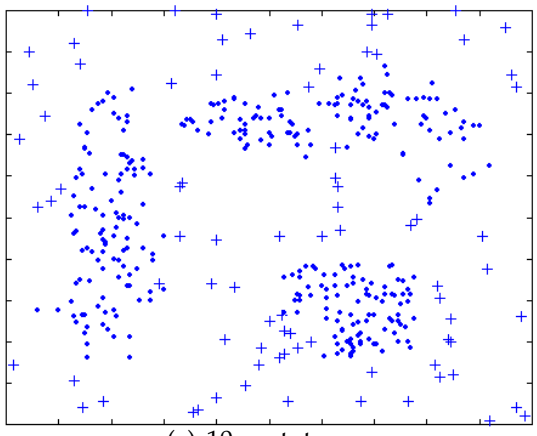
FCM provides membership degrees for each feature vector to every cluster. One approach could be, to assign a feature vector to the corresponding



(a) 1 prototype



(b) 3 prototypes



(c) 10 prototypes

Figure 3.15: Outlier detection with different number of prototypes

Cluster	mean flight duration (s) (before outlier test)	RMSE	mean flight duration (s) (after outlier test)	RMSE
1	2021.18	266.17	2021.18	266.17
2	2497.13	407.90	2465.71	358.68
3	2136.85	268.93	2136.85	268.93
4	2303.41	409.35	2303.41	409.35
5	2186.22	292.04	2186.22	292.04
6	1872.23	180.45	1872.23	180.45
7	2033.31	395.33	2033.31	395.33
8	1879.28	187.12	1879.28	187.12
9	1839.65	90.95	1839.65	90.95
10	2566.15	517.01	2523.28	492.60

Table 3.1: Estimated flight duration before and after outlier treatment

clusters with the two highest membership degrees. The feature vector is considered as an outlier if the algorithm comes to the same decision for both clusters. In cases where the algorithm gives no definite answers, the feature vector can be labelled and processed by further analysis.

3.9.2 Eliminating Outlying Weather Data

We applied the above method on a weather data set describing the weather situation at Frankfurt Airport at 12:20 PM. Partitioning the weather data is done using k-means with 10 prototypes. Since the weather data set is high-dimensional in the feature space we prescind here from showing a visualization of the clustering results. Though, table 3.1 shows some numerical results for the flight duration before the outlier treatment and afterwards.

The proposed outlier procedure removes a total of four outliers in two clusters. The according clusters are highlighted in the table by means of light grey background. Indeed, both clusters benefit from removing the outliers insofar that the estimation of the flight duration, using a simple measure like the mean, can be improved to a considerable extent. The lower RMSE for the flight duration estimation in both clusters confirms this.

4 Fuzzy Classification Rules

Classification aims at associating input data to predefined classes. Many techniques, such as neural networks and support vector machines (SVM), have been successfully applied for this purpose. Unfortunately, these techniques have the disadvantage of being hardly interpretable. The user of a suchlike technique cannot see why the system behaves like it does. Available expert knowledge cannot be easily integrated into the system.

Both these issues are not the case for fuzzy systems [59]. Fuzzy rules can be understood by a user and own rules can be integrated if necessary. Typically, fuzzy rules describe an inference scheme:

$$\mathcal{R}: \quad \texttt{if} \quad \textit{antecedent} \quad \texttt{then} \quad \textit{consequent}$$

where the antecedent is described by the input variables:

$$x_1 \texttt{ is } A_1 \quad \texttt{and} \quad \ldots \quad \texttt{and} \quad x_l \texttt{ is } A_l$$

and the consequent by a single output variable y `is` B[6]. Input variables are defined by means of membership functions.

Instead of constructing an entire rule base by hand, one can automatically derive rules from data. Fuzzy rules are usually obtained from fuzzy clusters by projecting the clusters to the coordinate spaces, but also various other techniques are commonly used [5, 34, 50]. Despite the good interpretability of single fuzzy rules, the analysis of an entire fuzzy rule base can be exhausting. Particularly if the data comprehend many attributes, i.e. the input data is high-dimensional, interpretation becomes difficult. The following section therefore discusses a technique to visualize fuzzy rules and the data set classified by the rule system as well. We have presented this approach in [83, 85].

[6] There are also other concepts of fuzzy rules, e.g. where the rule consequent is employed as a linear function of the input variables, which are not considered here.

4.1 Rule Classification Visualization

To start with, we follow the terminology of fuzzy rules according to the definition that is given in [5, 32]. Later we will apply the rule base visualization approach on a rule base that is generated by a more general rule construction technique.

A trapezoidal membership function of a fuzzy rule is defined by four parameters $< a_i, b_i, c_i, d_i >$ (see figure 4.1). The rule's *core region* for attribute i is defined by parameter b_i and c_i. It describes the region of the membership function that is supported by training examples during the rule learning phase. The rule's *support region* for attribute i is defined by parameter a_i and d_i. The support region might be constrained as the figure shows, but also open to $\pm\infty$ depending on the training algorithm. In addition, a centre vector of each rule can be determined by means of the core region's centre for each attribute of the rule.

Further, we define neighbourhood of rule centre vectors according to overlap regarding the core regions of the rule system. Neighbourhood N_{ef} of rule $\mathcal{R}_e$ and $\mathcal{R}_f$ can either be 1 if all core regions of both rules overlap, or 0 if not.

Combining the centre vectors and the neighbourhood N_{ef} we can define a dissimilarity (or distance) matrix D as it is used with Sammon's mapping that we already described in section 2.1. Sammon's mapping finds a low-dimensional representation of high-dimensional data trying to preserve distances between feature vectors. Accordingly, we can use Sammon's mapping to find a two- or three-dimensional mapping of the rule centre vectors (or their representation by means of the dissimilarity matrix). Thus, we use the normalized rule centre vectors to determine the required distance matrix and enlarge the distance of non-neighbouring rules by N_{ef}. Of course, no guarantee can be given, that all neighbouring rules will be placed appropriately, but considering core based neighbourhood might improve the chance to obtain feasible transformations.

Once the rule centre vectors are mapped in the plane or the 3D-space, we propose to place the data set's feature vectors according to their membership

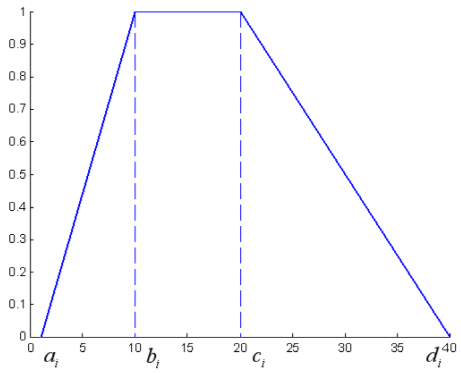


Figure 4.1: A trapezoidal membership function

degree to the two rules that yield the highest response. Thus, we place the feature vectors proportional to both rule centre vectors. A visualization like this reveals some interesting aspects of the rule system:

- Similar rules and neighbouring rules can be visualized by their distance and a drawn link, respectively.
- Classified feature vectors symbolize by their colour and their proportional distance to the respective rule centre vector which rule fires to what degree.
- Misclassified feature vectors can be detected when using appropriate symbols or colours for them.
- Conflicting rules (visualized by connected rule centres of different colours) can be identified.

In the next section we will demonstrate the proposed technique on some benchmark examples.

4.1.1 Illustrative Examples

Figure 4.2 shows an exemplary rule classifier that is learned based on the well known Wine data set (see section 2.4.3). We applied the fuzzy rule learning algorithm as described in [5] and obtained ten rules which classify the entire

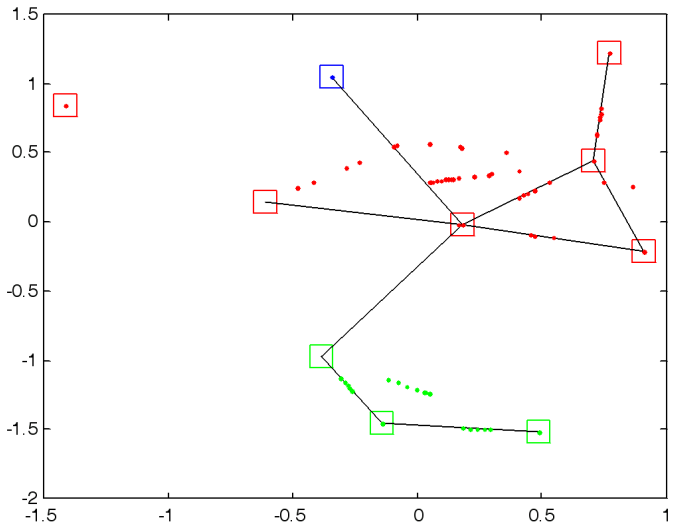


Figure 4.2: 2D-Visualization of an example for a rule classifier on the Wine data set

data set correctly. Rule centre vectors are visualized by squares (□). Connections between rule centre vectors indicate their neighbourhood regarding the core region. Rules of the same class are visualized by the same colour. Additionally, data objects are visualized by means of points. A feature vector's membership to a certain rule can be differentiated by means of its colour and its distance to rule centre vectors.

The figure reveals some interesting facts. In consequence of placing vectors in the plane depending on their membership degree to the two rules that yield the highest response, classified feature vectors will be placed on an imaginary line that connects two rule centre vectors. Note, feature vectors may not only be represented by neighbouring rules corresponding to the core based neighbourhood definition whose neighbourhood is visualized by lines in the figure. As the figure reveals, for some neighbouring rules the data set contains no data that lie in the core regions of those rules. Two of ten rules represent data that lie not in any of the core regions of these rules. If two rules yield similar membership degrees to a feature vector, it will be placed in the middle between these rule centre vectors. Of course, the classification that will be done in such cases is not that confiding since the decision comes

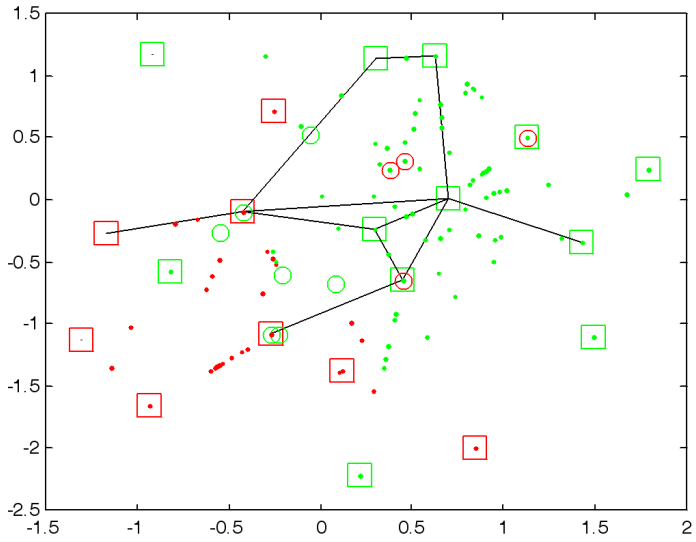
randomly if no further information is available.

This classifier is almost ideal. Despite of one conflicting red rule that overlaps with one blue rule and one green rule no misclassifications occur. There are also no rule pairs representing different classes that compete for some data. Actually, the visualization tool can provide much more insight into classifiers that comprise problematic aspects. The following examples will demonstrate some more prospects of this technique.

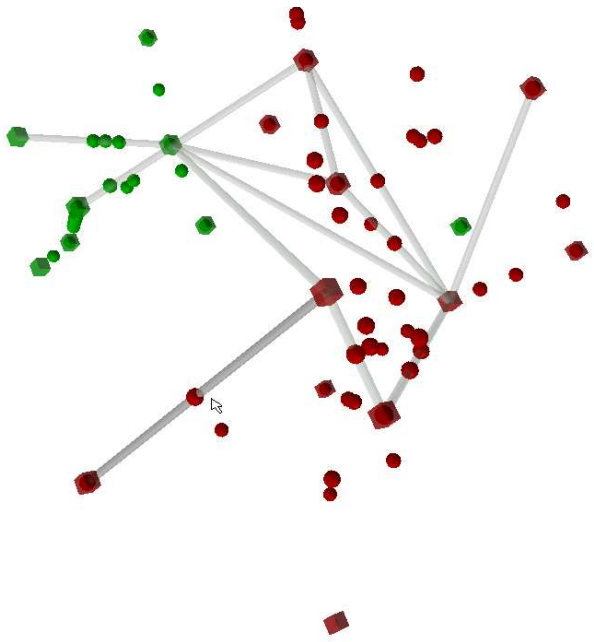
The second example is the Wisconsin breast cancer data set [64]. Each patient in the database had a fine needle aspirate taken from her breast. Then nine attributes where determined and analyzed to discriminate benign from malignant breast lumps. Figure 4.3(a) shows the visualization of the learned rule system and the corresponding classified data set. For this example we divided the data set into a training data set and a test data set by choosing randomly 50% of the data for each of both data sets. We used the training data set to learn the fuzzy rule classifier. The test data set was applied on the learned classifier which yields the figure above using the proposed visualization technique.

The figure shows clearly that rule centre vectors which represent the same class are mapped in the same region in the plane. There are two neighbouring rules that represent different classes. These rules misclassify some feature vectors. Some rules respond only with small membership degrees to few feature vectors and do not yield high response to any other feature vector. This fact is shown in the figure by rule centre vectors that have no adjacent feature vectors. Thus, the figure reveals that the rule system can be pruned here.

Figure 4.3(b) shows a 3-dimensional visualization of the rule classifier on the training data of the Wisconsin breast cancer data set. Feature vectors of different classes are visualized by small spheres of different colours. Rule centres are visualized by cubes. Transparency helps to identify feature vectors which are positioned exactly on the same coordinate as rule centres. Light grey connections between rule centres indicate rule neighbourhood. Three-dimensional visualization is mainly efficient when interaction (zooming, rotating, etc.) is provided. The figure results from a Java3D implemen-



(a) 2D-Visualization of an examle for a rule classifier on the Wisconsin breast cancer data set



(b) 3D-Visualization of an examle for a rule classifier on the Wisconsin breast cancer data set

Figure 4.3: Visualization of rule classifiers on the Wisconsin breast cancer data set

tation that enables the user to interact. In the foreground of the figure a dark grey connection can be found. In the actual implementation, feature vectors can be clicked and the two rules that yield the highest response to the feature vector will be visualized by a dark grey connection. Clicking the same feature vector again causes the disappearance of the according connection. This feature helps the analyst to identify interesting rules and feature vectors as well.

4.1.2 Visualization of Classification Rules for Flight Duration Prediction Based on Weather Data

The impact of weather on flight duration of arriving aircraft has been analyzed in various studies [61, 76, 78]. Several techniques, such as neural networks, support vector machines, linear regression and regression trees have been applied to the data [11, 65, 74, 75].

Figure 4.4 shows an example of a rule classifier that was learned on a sample of the weather data using the method described in [5]. As mentioned earlier, the weather data set consists of weather reports which are regularly released every thirty minutes and describe the weather situation at Frankfurt Airport. To demonstrate our visualization technique that is suitable mainly for smaller data sets or rule bases respectively, we consider here only the weather reports given at 12:20 PM for the year 1998. Due to some missing weather reports for the considered time period the data set comprehends 333 data. The flight duration times were grouped into three classes: short flights with one minute delay with respect to the average flight duration in the TMA, medium flights with eight minutes delay and long flights with even more delay.

A rule classifier was learned using this data set and pruned to eleven rules. For a better interpretability some additional lines are plotted (light grey) to visualize which rule pair yields the highest response to a certain weather report that could not be classified correctly. Solid black lines indicate overlapping rules again. It can be observed that rules covering short flights (green rule centre vectors) and medium flights (blue rule centre vectors) overlap

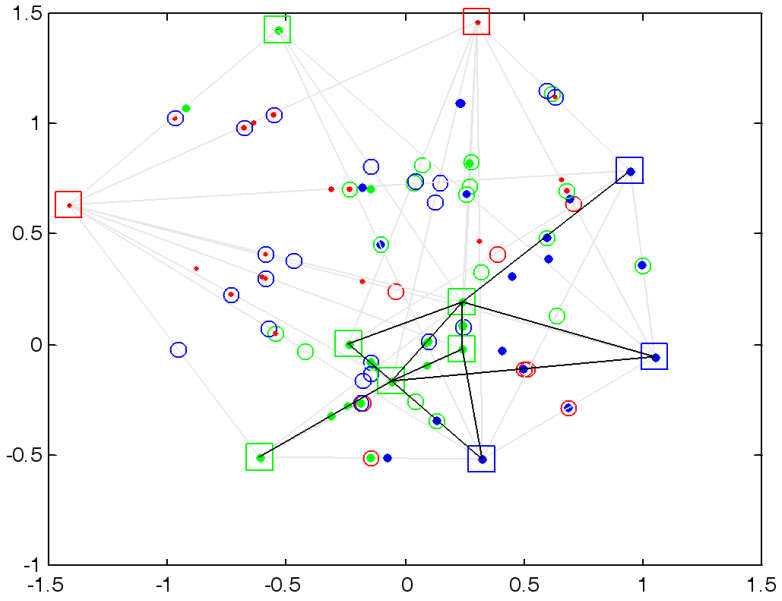


Figure 4.4: 2D-Visualization of an exemplary rule classifier on the weather data

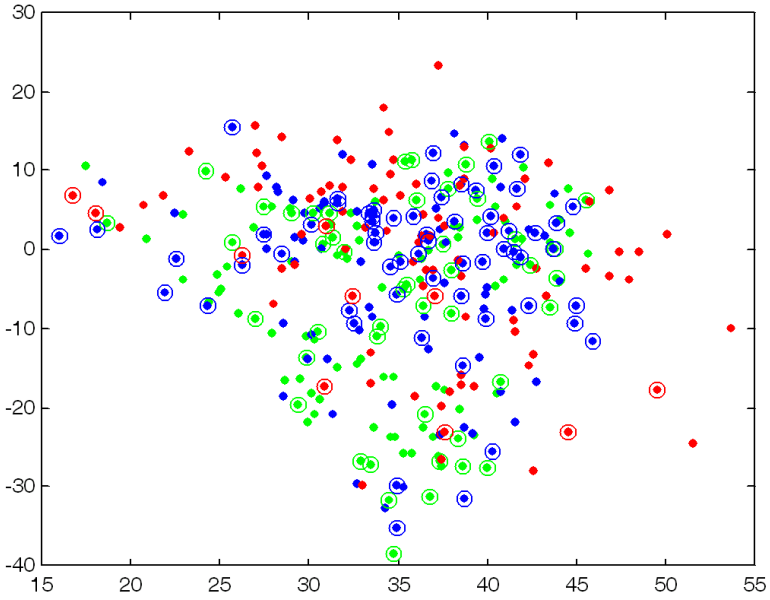


Figure 4.5: POLARMAP-Visualization of the weather data set

sometimes. Rules that cover long flight durations (red rule centre vectors) are fairly distant to other rules in the figure which emphasizes the impact of weather on flight duration. The proximity of rules that cover short flights, reflects the similarity of those weather conditions. Of course, rules, covering medium flights, are sometimes alike to short flight rules but also to long flight rules. This is reflected by the rule centres' positions in the figure. If only one rule covers a certain weather report the feature vector will be placed directly on the rule centre vector. Thus, it cannot be inspected on the graph, how many feature vectors overlap. Further development should focus this problem. Due to pruning 118 weather reports cannot be classified correctly. These weather reports are depicted by the circles in the figure.

Figure 4.5 shows a 2-dimensional mapping of the weather data set using POLARMAP. Data points surrounded by a circle correspond to misclassified weather reports. The figure reveals why this data is not covered by any rule of the pruned rule system. Mostly these points are located in areas where all classes of weather reports appear. In these cases the rule learner has to use many rules to classify the data set correctly. Since the pruning strategy simply removes those rules which cover only few weather reports, these points cannot be classified correctly thereafter. The figure also gives some hints for the partly low classification rates (green: 61%, blue: 48%, red: 89%). The discretization of the continuous flight duration times to three flight duration classes generates numerous misclassifications especially on class borders. Further analysis should investigate a suitable binning. Binning could be improved using histograms (e.g. see figure 4.6) to find an appropriate discretization of the flight duration attribute. As it can be seen in figure 4.6, there are two significant gaps at 1900s and 2150s that might be a better choice to use for discretization.

The rule construction technique as discussed in section 4.1 assumes that for each rule a centre vector can be computed. Indeed, the aforesaid technique always generates membership functions for each variable of the data set in each rule. However, other rule construction algorithms try to find rules that use a minimal number of variables to describe the classification task. Differ-

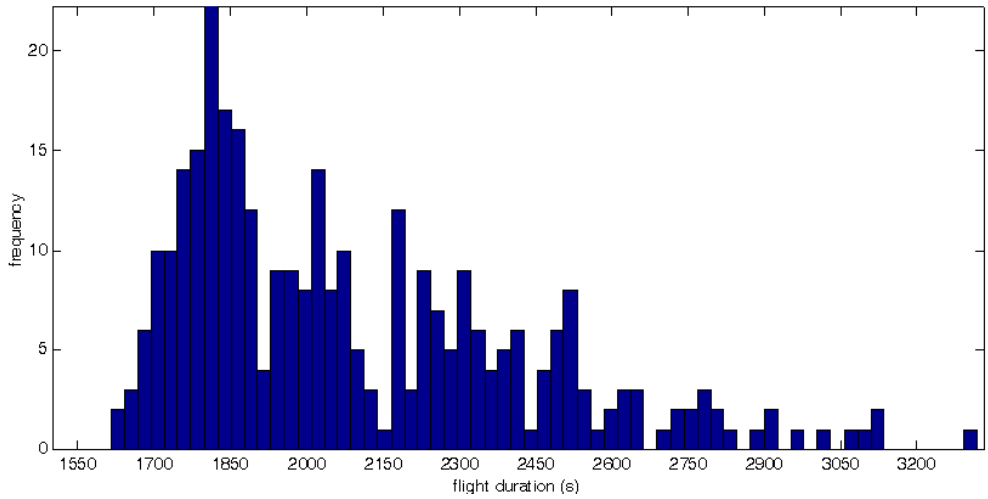
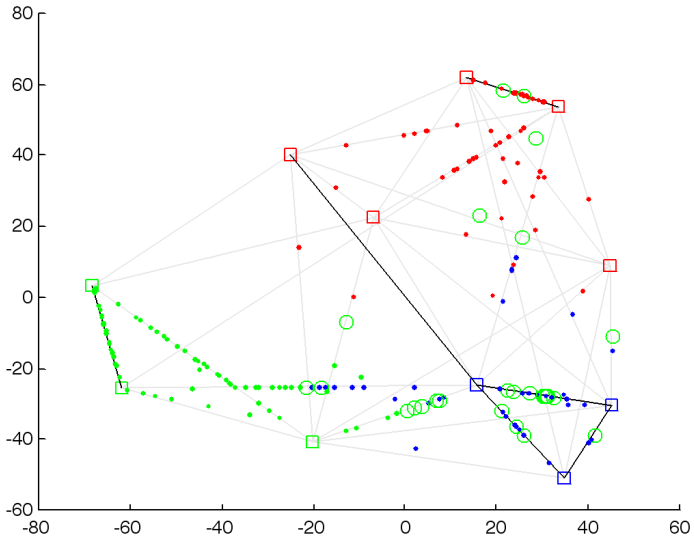


Figure 4.6: Histogram of flight durations

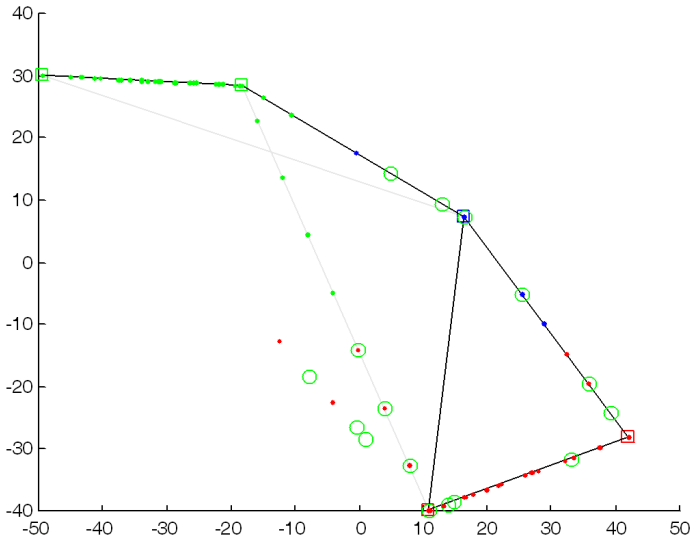
ent rules of a rule base consider different variables. Consequently, rule centre vectors that we have used so far to build a distance matrix for the Sammon's mapping procedure cannot be determined directly.

To build up a distance matrix (or generally a dissimilarity matrix) distances are not essentially needed. Dissimilarity can be defined by comparing single membership functions of rule pairs. Starting with an initially zero valued dissimilarity matrix, dissimilarity of a rule pair has to be augmented if membership functions for the same variable do not overlap. Dissimilarity of a rule pair that is not overlapping or that predicts different classes can also be augmented. A suchlike derived dissimilarity matrix can be straightly used with Sammon's mapping.

Figure 4.7 shows two rule bases that are generated by the NEFCLASS fuzzy rule learner [68]. The rule base that is visualized in the upper figure is built using the same class definition as described earlier in this section. The classification rate is comparable to the rule classifier that we have discussed in this context. For a better readability only misclassified short flights are depicted in the figure. The visualization reveals that the rule base contains some overlapping rules – rules that get non-zero membership degrees to a shared subset of data of the same class – but also two conflicting rules that



(a) A rule classifier with 11 rules



(b) A rule classifier with 5 rules

Figure 4.7: 2D-Visualization of rule classifiers on the weather data including some misclassified short flights (Sammon's mapping based on a dissimilarity matrix)

get non-zero membership degrees to a shared subset of data of a different class. Eye-catching are some misclassified short flights whose two highest membership degrees get rules that actually classify long flights. Investigations on the raw traffic data have shown that the flight durations for these flights where originally missing. These missing values were replaced by the mean value which obviously does not suitably reflect the reality. There are also some short flights that get high membership degrees to rules that classify medium and long flights. Despite of bad weather conditions and demanding traffic at the airport these aircraft could land very fast.

Figure 4.7(b) shows the mapping of a rule base that is composed by five rules only. The reduction of the rule number is an effect that arises when taking the flight duration frequencies (as shown in figure 4.6) into account. Flight duration classes are defined according to the gaps in the histogram at 1900s (as the upper bound for short flights) and at 2150s (as the upper bound for medium flights)[7]. The classification rate is approximately the same as for the rule base with 11 rules. As the figure reveals, this rule base contains more conflicting rules. Since the class definition has changed the intra class classification rate cannot be compared directly with the prior fuzzy classifier.

[7] Using the gap at 2450s as the upper bound for medium flights does not improve the fuzzy classifier.

5 Conclusions

In this work we have contributed to the field of data mining and data visualization against the background of air traffic management tasks. We have presented some new visualization techniques as well as some suitable extensions to well-known methods.

Visualization of high-dimensional data is an active research area. Multidimensional scaling aims at finding a low-dimensional representation of high-dimensional data while preserving similarity of objects. We focused in this thesis on the development of two new MDS-related techniques, namely MDS_{polar} and POLARMAP, that provide some valuable aspects. Besides computational efficiency, both approaches allow to map new data that has not been used to learn the model. So far only few approaches support this feature while being computational expensive. In contrast to conventional MDS approaches were distances or dissimilarities will be preserved, our transformation bases in both approaches on the preservation of angles between feature vectors when mapping high-dimensional data onto the plane. MDS_{polar} provides the possibility to map new data that has not been used to learn the model, since the solution is described by means of a system of linear equations. POLARMAP's solution is even more comfortable because a function is learned that can be applied for new data objects. Similar to kernel methods POLARMAP can implicitly represent the data in a new feature space to improve the transformation. The application of these visualization techniques revealed interesting correlations between weather and flight durations at airports. With density-based multidimensional scaling we have introduced another approach that combines conventional MDS and density preservation successfully.

Clustering is the process of grouping data into several clusters containing similar objects. The task of clustering is to maximize the intraclass similarity

and minimize the interclass similarity. Many clustering techniques are fairly sensitive to noisy data. The effect is that prototypes, centre vectors that represent the clusters, will be placed on suboptimal positions in the feature space. Noise clustering is a common technique that robusts well-known clustering algorithms, like fuzzy *c*-means. However, noise clustering depends strongly on the number of prototypes used for the clustering which is not covered by the original noise clustering approach. We provided in this work a technique that estimates the respective control parameter - the noise distance - such that noise clustering becomes widely robust to the prototype number. Further we discussed a new technique that combines clustering and statistical outlier detection. On the basis of the weather data we have shown that the treatment of outliers by means of our technique yields more accurate results for the flight duration prediction.

Validation of clustering partitions is a crucial step to check whether the prototypes fit the data clusters. To date, plenty of validity indices exist condensing the clustering result to a single value to rate the partition quality. We have also discussed some visual techniques to validate clustering partitions. With the approach called single clustering visualization we have presented a new technique to visualize a clustering partition from the perspective of a certain cluster. Interesting aspects become visual, such as compactness of clusters, the existence of outliers and whether the number of prototypes is chosen appropriately or not.

Furthermore, we provided a very efficient technique to visualize fuzzy rule classifiers. Fuzzy rules a known to be easily interpretable. Therefore they are often used to classify data. Our 2D- or 3D-visualizations of rule bases reveal important aspects and thusly improve interpretability. The application of this visualization technique enabled us to identify misclassified data as well as outliers and helped us to identify conflicting and overlapping rules for the combined weather and flight duration data set.

The application of the different visualization and data preprocessing techniques has shown that there is still some work to do. Our multidimensional scaling techniques might sometimes have a large constant in the computa-

tion scheme which restrains their efficiency for large data sets. To cope with this problem we have discussed the binning issue that allows to consider relevant data pairs only. Further investigations should concern the criterion for data that belong to the same bin and finding the appropriate bin size to improve transformation quality and efficiency. Of course, visualization can only reflect existing underlying data structures. Consequently, the proposed techniques are fairly dependent on the provided data. The interpretation of the obtained results cannot completely substitute expert knowledge. Finally, visualization with scatter plots and related methods is restricted to the displaying medium. Thus, visualization of very large data sets demands sophisticated techniques to overcome such limitations. Future work should concern these aspects.

Aside from improving the prediction accuracy by means of data cleaning we have to declare that ultimate accuracy cannot be achieved since deviations in air traffic occur naturally. The application of support vector regressors on this data have shown that taking non-linearities into account may improve prediction accuracy significantly which proves that there is still room for advancement at the expense of interpretability. Visualization is a very important medium to transmit knowledge to experts and decision makers and should be elaborated in future particularly for powerful and hardly interpretable techniques.

List of Figures

List of Algorithms

A Mappings of the Weather Data

The following figures show the effect of the bin size on MDS$_{polar}$-mappings of a weather data set that contains 3510 data[8]. When using smaller bin sizes (say 50 or 100) the mapping already reveals some of the characteristics that can be observed for mappings with larger bin sizes. Accordingly, short flights (green points) are spread over the entire feature space, whereas medium flights (blue points) and long flights (red points) are mainly represented in a separate area. A stable mapping will be obtained when using a bin size of 200 or higher.

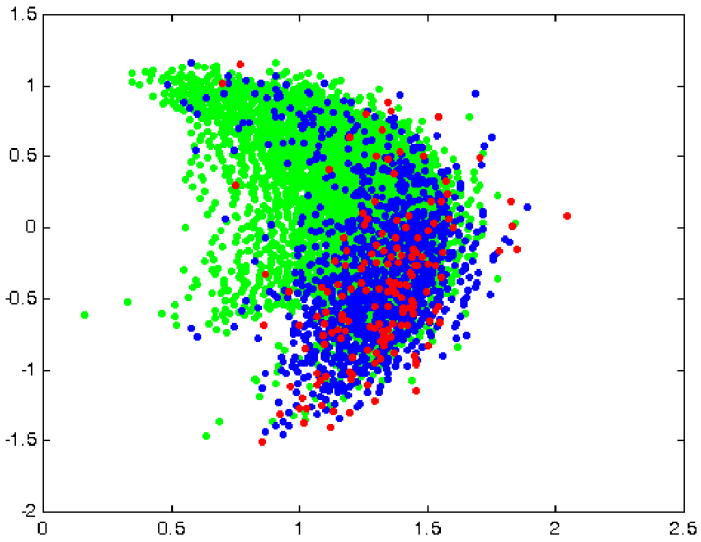


Figure A.1: Mapping of the weather data with MDS$_{polar}$ (bin size=50)

[8]Due to its similar approach the effect of the bin size on mappings with POLARMAP is comparable with MDS$_{polar}$.

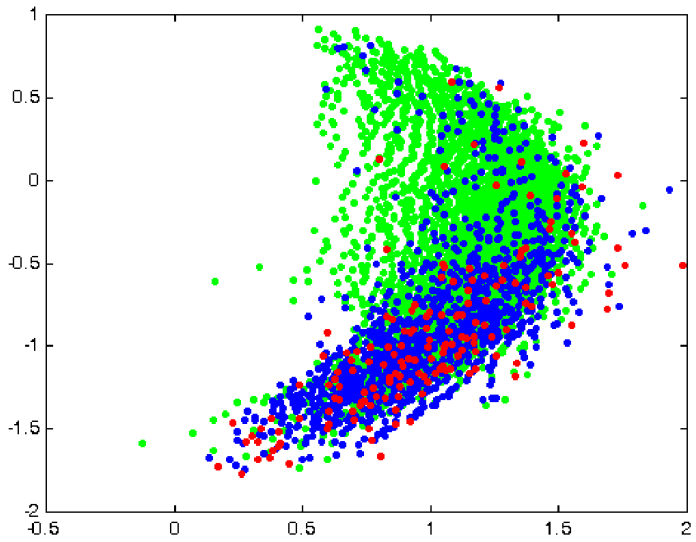


Figure A.2: Mapping of the weather data with MDS_{polar} (bin size=100)

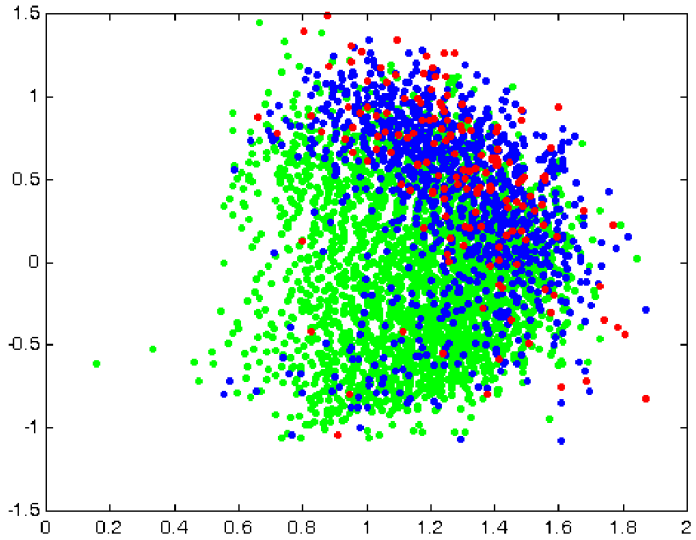


Figure A.3: Mapping of the weather data with MDS_{polar} (bin size=200)

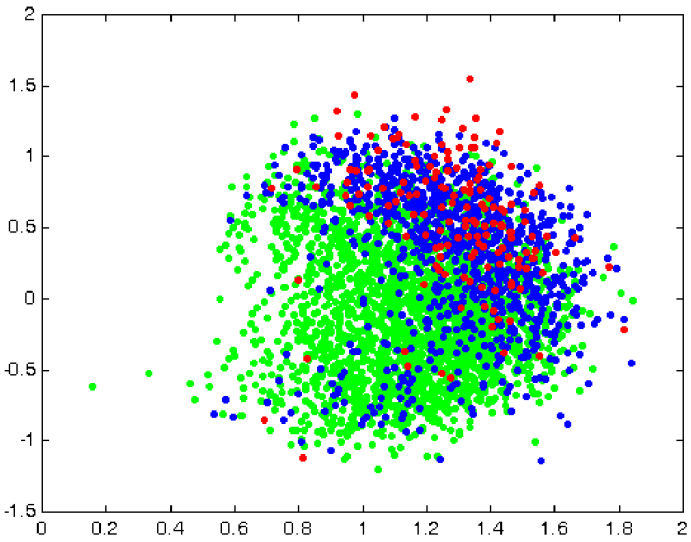


Figure A.4: Mapping of the weather data with MDS_{polar} (bin size=500)

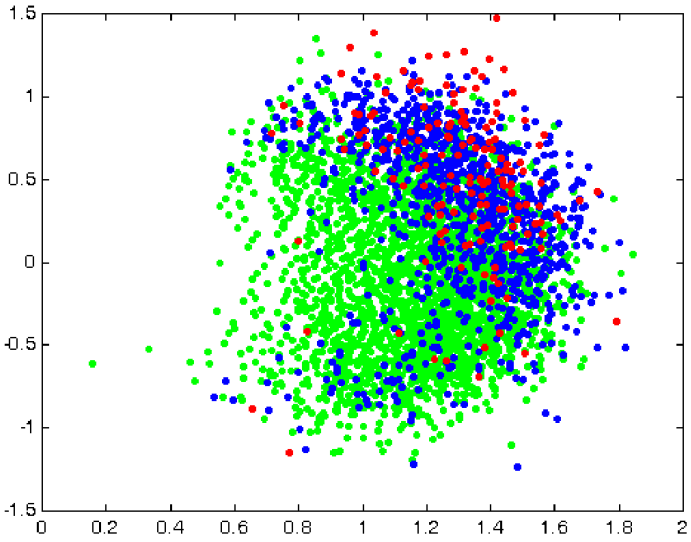


Figure A.5: Mapping of the weather data with MDS_{polar} (bin size=1000)

Bibliography

[1] Allan, S.S., Beesley, J.A., Evans, J.E., Gaddy, S.G.: Analysis of delay causality at Newark International Airport. 4th USA/Europe Air Traffic Management R&D Seminar, Santa Fe, 2001.

[2] Allan, S.S., Gaddy, S.G.: Delay reduction at Newark International Airport using terminal weather information systems. In Proceedings of the 9th Conference on Aviation, Range, and Aerospace Meteorology, Orlando, FL, 141–146, 2000.

[3] Barnett, V. Lewis, T.: Outliers in statistical data. John Wiley & Sons, New York, 1994.

[4] Berthold, M.R.: Fuzzy models and potential outliers. In Dave, R.N., Sudkamp, T. (eds) Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS 1999), 532–535, IEEE Press, 1999.

[5] Berthold, M.R.: Mixed fuzzy rule formation. International Journal of Approximate Reasoning, 32, 67–84, Elsevier, 2003.

[6] Bezdek, J.C.: A convergence theorem for the fuzzy ISODATA clustering algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2, 1–8, 1980.

[7] Bezdek, J.C.: Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York, 1981.

[8] Bezdek, J.C., Hathaway, R.J.: VAT: a tool for visual assessment of (cluster) tendency. In Proceedings of International Joint Conference on Neu-

ral Networks (IJCNN 2002), IEEE Press, Piscataway, NJ, 2225–2230, 2002.

[9] Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. IEEE Transactions on Systems, Man and Cybernetics, 28(3), 301–315, 1998.

[10] Borg, I., Groenen, P.: Modern multidimensional scaling : theory and applications. Springer, Berlin, 1997.

[11] Breiman, L., Fiedmann, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Chapman & Hall, New York, 1984.

[12] Breunig, M., Kriegel, H-P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In Chen, W., Naughton, J.F., Bernstein, P.A. (eds) Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2000), Dallas, TX, 93–104, 2000.

[13] Callaham, M.B., DeArmon, J.S., Cooper, A.M., Goodfriend, J.H., Moch-Mooney, D., Solomos, G.H.: Assessing NAS performance: normalizing for the effects of weather. 4th USA/Europe Air Traffic Management R&D Symposium, Santa Fe, 2001.

[14] Chalmers, M.: A linear iteration time layout algorithm for visualising high-dimensional data. In Yagel,R., Nielson, G.M. (eds) Proceedings of IEEE Visualization 1996, San Francisco, CA, 127–132, 1996.

[15] Chaudhuri, B.B., Bhowmik, P.R.: An approach of clustering data with noisy or imprecise feature measurement. Pattern Recognition Letters, 19, 1307–1317, 1998.

[16] Chen, Z.: Data mining and uncertain reasoning. John Wiley & Sons, New York, 2001.

[17] Dave, R.N.: Characterization and detection of noise in clustering. Pattern Recognition Letters, 12, 657–664, 1991.

[18] Dave, R.N., Krishnapuram, R.: Robust clustering methods: a unified view. IEEE Transactions on Fuzzy Systems, 5, 270–293, 1997.

[19] Dave, R.N., Sumit, S.: Generalized noise clustering as a robust fuzzy c-m-estimators model. Conference of the North American Fuzzy Information Processing Society (NAFIPS 1998), Pensacola Beach, FL, 256–260, 1998.

[20] Dave, R.N., Sumit, S.: On generalizing the noise clustering algorithms. In Proceedings of the 7th International Fuzzy Systems Association World Congress (IFSA 1997), 3, 205–210, 1997.

[21] Davies, D.L., Bouldin, W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1, 224–227, 1979.

[22] Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. Journal of Cybernetics, 4, 95–104, 1974.

[23] van Eck, N.J., Frasincar, F., van den Berg, J.: Visualizing concept associations using concept density maps. In IEEE Proceedings of the 10th International Conference on Information Visualisation (IV'06), London, 270–275, 2006.

[24] van Eck, N.J., Waltman, L., van den Berg, J.: A novel algorithm for visualizing concept associations. In IEEE Proceedings of the 16th International Workshop on Database and Expert Systems Applications, 405–409, 2005.

[25] Estivill-Castro V., Yang J.: Fast and robust general purpose clustering algorithms. Data Mining and Knowledge Discovery, 8(2), 127–150, Springer, Netherlands, 2004.

[26] Evans, J.E., Allan, S., Robinson, M.: Quantifying delay reduction benefits for aviation convective weather decision support systems. In Proceedings of the 11th Conference on Aviation, Range, and Aerospace Meteorology, Hyannis, 2004.

[27] Faloutsos, C., Lin, K.: Fastmap: A fast algorithm for indexing, datamining and visualization of traditional and multimedia data sets. In

Carey, M.J., Schneider, D.A. (eds) Proceedings of ACM SIGMOD International Conference on Management of Data, San Jose, CA, 163–174, 1995.

[28] Fayyad, U., Grinstein, G.G., Wierse, A.: Information visualization in data mining and knowledge discovery. Morgan Kaufmann Publishers, San Francisco, CA, 2002.

[29] Feil, B., Balasko, B., Abonyi, J.: Visualization of fuzzy clusters by fuzzy sammon mapping projection: application to the analysis of phase space trajectories. Soft Computing Journal, Springer, Berlin/Heidelberg, (to appear).

[30] Frigui, H., Krishnapuram, R.; A robust competitive clustering algorithm with applications in computer vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5), 450–465, 1999.

[31] Fukuyama, Y., Sugeno, M.: A new method of choosing the number of clusters for the fuzzy c-means method. In Proceedings of the 5th Fuzzy System Symposium, 247–250, 1989.

[32] Gabriel, T.R., Thiel, K., Berthold, M.R.: Rule visualization based on multi-dimensional scaling. In Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2006), 66–71, 2006.

[33] Gath I., Geva, A.B.: Unsupervised optimal fuzzy clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11, 773–781, 1989.

[34] Genther, H., Glesner, M.: Automatic generation of a fuzzy classification system using fuzzy clustering methods. In Proceedings of the ACM Symposium on Applied Computing (SAC 1994), 180–183, Phoenix, 1994.

[35] Grubbs, F.: Procedures for detecting outlying observations in samples. Technometrics, 11(1), 1–21, 1969.

[36] Gustafson, D.E., Kessel, W.C.: Fuzzy clustering with a fuzzy covariance matrix. In Proceedings of the IEEE Conference on Decision and Control, San Diego, 761–766, 1979.

[37] Hathaway, R.J., Bezdek, J.C.: Visual cluster validity for prototype generator clustering models. Pattern Recognition Letters, 24, 9–10, 2003.

[38] Hawkins, D.: Identification of outliers. Chapman & Hall, London, 1980.

[39] Hand, D, Mannila, H., Smyth, P.: Principles of data mining (Adaptive computation and machine learning). Bradford Book, Cambridge, MA, 2001.

[40] Hansen, M., Bolic, T.: Normalization of airport and terminal area operational performance: a case study of Los Angeles International Airport. 4th USA/Europe Air Traffic Management R&D Seminar, 2001.

[41] Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning (data mining, inference, and prediction). Springer, New York, 2003.

[42] Hoffman, P.E., Grinstein, G.G.: A survey of visualizations for high-dimensional data mining. In: Fayyad, U., Grinstein, G.G., Wierse, A. (eds) Information visualization in data mining and knowledge discovery. Morgan Kaufmann Publishers, 47–82, San Francisco, CA, 2001.

[43] Hoffman, P.E., Grinstein, G.G., Marx, K., Grosse, I., Stanley, E.: DNA visual and analytic data mining. In Proceedings of the IEEE Conference on Visualization, 437–441, Phoenix, 1997.

[44] Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy cluster analysis. John Wiley & Sons, Chichester, 1999.

[45] Huband, J.M., Bezdek, J.C., Hathaway, R.J.: bigVAT: visual assessment of cluster tendency for large data sets. Pattern Recognition Letters, 38, 1875–1886, 2005.

[46] Jerding, D.F., Stasko, J.T.: The information mural: a technique for displaying and navigating large information spaces. In Proceedings of the IEEE Visualization Symposium of Information Visualization, 43–50, Atlanta, 1995.

[47] Keller, A.: Fuzzy clustering with outliers. In: Whalen, T. (ed) Peach-Fuzz 2000, 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2000), 143–147, Atlanta, 2000.

[48] Klawonn, F.: Noise clustering with a fixed fraction of noise. In: Lotfi, A., Garibaldi, J.M. (eds) Applications and Science in Soft Computing, Springer, Berlin, 133–138, 2004.

[49] Klawonn, F., Chekhtman, V., Janz, E.: Visual inspection of fuzzy clustering results. In: Benitez, J., Cordon, O., Hoffmann, F., Roy, R. (eds) Advances in Soft Computing: Engneering Design and Manufacturing. Springer, London, 65–76, 2003.

[50] Klawonn, F., Kruse, R.: Constructing a fuzzy controller from data. Fuzzy Sets and Systems, 85(2), 177–193, 1997.

[51] Klawonn, F., Rehm, F.: Clustering techniques for outlier detection. In: Wang, J. (ed) Encyclopedia of Data Warehousing and Mining. Idea Group, Hershey, 180–183, 2006.

[52] Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In Gupta, A., Shmueli, O., Widom, J. (eds) Proceedings of the 24th International Conference Very Large Data Bases (VLDB'98), Morgan Kaufmann Publishers, San Francisco, CA, 392–403, 1998.

[53] Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-based outliers: algorithms and applications. The VLDB Journal - The International Journal on Very Large Data Bases, 8(3–4), 237–253, 2000.

[54] Kohonen, T.: Self-organizating maps. Springer, New York, 1997.

[55] 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, 177–188, 2002.

[56] Krishnapuram, R. Freg, C.: Fitting an unknown number of lines and planes to image data through compatible cluster merging. Pattern Recognition, 25, 385–400, 1992.

[57] Krishnapuram, R., Keller, J.M.: A possibilistic approach to clustering. IEEE Transactions on Fuzzy Systems, 1, 98–110, 1993.

[58] Krishnapuram, R., Keller, J.M.: The possibilistic c-means algorithm: insights and recommendations. IEEE Transactions on Fuzzy Systems, 4, 385–393, 1996.

[59] Kruse, R., Gebhardt, J., Klawonn, F.: Foundations of Fuzzy Systems. John Wiley & Sons, Chichester, 1996.

[60] Kruskal, J.B., Wish, M.: Multidimensional scaling. SAGE Publications, Beverly Hills, 1978.

[61] Lesot, M-J., Rehm, F., Klawonn, F., Kruse, R.: Prediction of aircraft flight duration. In Proceedings of the 11th IFAC Symposium on Control in Transportation Systems, Delft, 107–112, 2006.

[62] Lowe, D., Tipping, M.E.: Feed-forward neural networks topographic mapping for exploratory data analysis. Neural Computing and Applications, 4(2), 83–95, 1996.

[63] Mahalanobis, P.C.: On the generalized distance in statistics. In Proceedings of the National Institute of Science of India, 12, 49–55, 1936.

[64] Mangasarian, O.L., Wolberg, W.H.: Cancer diagnosis via linear programming. SIAM News, 23(5), 17–18, 1990.

[65] Mitchell, T.M.: Machine Learning. McGraw-Hill, New York, 1997.

[66] Morrison, A., Chalmers, M.: A pivot-based routine for improved parent-finding in hybrid MDS. Information Visualization, 3, 109–122, 2004.

[67] Morrison, A., Ross, G., Chalmers, M.: Fast multidimensional scaling through sampling, springs and interpolation. Information Visualization, 2, 68–77, 2003.

[68] Nauck, D., Kruse, R.: Neuro-fuzzy classification with NEFCLASS. In: Bachem, A., Derigs, U., Fischer, D., Leopold-Wildburger, U., Möhring, R. (eds) Operations Research Proceedings, Springer, Berlin, 294–299, 1995.

[69] Naud, A.: Visualization of high-dimensional data using an association of multidimensional scaling to clustering. In Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems, 252–255, 2004.

[70] Naud, A.: An accurate MDS-based algorithm for the visualization of large multidimensional datasets. In Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J. (eds) Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2006), Lecture Notes in Computer Science 4029, Springer, 643–652, 2006.

[71] Nazeri, Z., Zhang, J.: Mining aviation data to understand impacts of severe weather on airspace system performance. In Proceedings of the International Conference on Information Technology: Coding and Computing, 518–523, 2002.

[72] Pal, N.R., Pal, K., Keller, J.M., Bezdek, J.C.: A possibilistic fuzzy c-means clustering algorithm. IEEE Transactions on Fuzzy Systems, 13(4), 517–530, 2005.

[73] Performance review commission: Performance Review Report. Eurocontrol, Brussels, 2005.

[74] Quinlan, J.R.: Induction of decision trees. Machine Learning, 1, 81–106, 1986.

[75] Rawlings, J.O., Pantula, S.G., Dickey, D.A.: Applied regression analysis. Springer, Berlin, 1998.

[76] Rehm, F.: Prediction of aircraft delays as a function of weather. 2nd WakeNet2-Europe Workshop, Langen, 2004.

[77] Rehm, F.; Klawonn, F.: Anwendung von Data-Mining-Methoden zur Vorhersage von Anflugverspätungen am Frankfurter Flughafen. 29th Annual Conference of the German Classification Society (GfKl), Magdeburg, 2005.

[78] Rehm, F., Klawonn, F: Learning methods for air traffic management. In Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005), Lecture Notes in Computer Science 3571, Springer, 992–1001, 2005.

[79] Rehm, F., Klawonn, F., Kruse, R.: Ausreißererkennung mit Fuzzy-Clustering-Methoden. In Mikut, R., Reischl, M. (eds) Proceedings of the 14. Workshop Fuzzy Systeme und Computational Intelligence, 3–12, 2004.

[80] Rehm, F., Klawonn, F., Kruse, R.: MDS_{polar} - a new approach for dimension reduction to visualize high dimensional data. In Famili, A.F., Kok, J.N., Peña, J.M., Siebes,A., Feelders, A.J. (eds) Proceedings of the 6th International Symposium on Intelligent Data Analysis (IDA 2005), Lecture Notes in Computer Science 3646, Springer, 316–327, 2005.

[81] Rehm, F., Klawonn, F., Kruse, R.: A novel approach to noise clustering for outlier detection. Soft Computing Journal, Springer, Berlin/Heidelberg, (to appear).

[82] Rehm, F., Klawonn, F., Kruse, R.: POLARMAP - a new approach to visualisation of high dimensional data. In IEEE Proceedings of the 10th International Conference on Information Visualisation (IV'06), London, 731–740, 2006.

[83] Rehm, F., Klawonn, F., Kruse, R.: Rule classification visualization of high-dimensional data. In Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU 2006), 1944–1948, 2006.

[84] Rehm, F., Klawonn, F., Kruse, R.: Single cluster visualization to optimize air traffic management. In Proceedings of the 30th Annual Conference of the German Classification Society (GfKl 2006), Springer, Heidelberg/Berlin, (to appear).

[85] Rehm, F., Klawonn, F., Kruse, R.: Visualization of fuzzy rule classifiers for flight duration forecast. In Proceedings of the Symposium on Fuzzy Systems in Computer Science (FSCS), Magdeburg, 2006.

[86] Rehm, F., Klawonn, F., Kruse, R.: Visualization of single clusters. In Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J. (eds) Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2006), Lecture Notes in Computer Science 4029, Springer, 663–671, 2006.

[87] Rehm, F., Klawonn, F., Kruse, R.: Visualizing single fuzzy c-means clusters. In Mikut, R., Reischl, M. (eds) Proceedings on the 15th Workshop on Computational Intelligence, 36–45, 2005.

[88] Rehm, F., Klawonn, F., Kruse, R.: Density-based multidimensional scaling. (submitted for publication).

[89] Rubens, M.: Fuzzy clustering algorithms and their cluster validity. European Journal of Operational Research, 10, 294–301, 1992.

[90] Runkler, T.A.: Information mining. Vieweg, Braunschweig/Wiesbaden, 2000.

[91] Sammon, J.W.: A nonlinear mapping for data structure analysis. IEEE Transactions on Computers, 18, 401–409, 1969.

[92] Santos-Pereira, C.M., Pires, A.M.: Detection of outliers in multivariate data: a method based on clustering and robust estimators. In: Härdle, W., Rönz, B. (eds) Proceedings of the 15th Symposium on Computational Statistics, Physica-Verlag, Heidelberg, 291–296, 2002.

[93] Sasse, M., Hauf, T.: A study of thunderstorm-induced delays at Frankfurt Airport, Germany. Meteorological Applications, 10, 21–30, 2003.

[94] Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press, 2004.

[95] Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science, 290, 2319–2323, 2000.

[96] Timm, H., Klawonn, F.: Different approaches for fuzzy cluster analysis with missing values. In Proceedings of the 7th European Congress on Intelligent Techniques & Soft Computing (EUFIT'99), 177–178, Aachen, 1999.

[97] Timm, H.: Fuzzy-Clusteranalyse: Methoden zur Exploration von Daten mit fehlenden Werten sowie klassifizierten Daten. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2002.

[98] Vapnik, V.N.: Statistical learning theory. John Wiley & Sons, New York, 1998.

[99] Williams, M., Munzner, T.: Steerable, progressive multidimensional scaling. In Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis 2004), Austin, TX, 57–64, 2004.

[100] Windham, M.P.: Cluster validity for fuzzy clustering algorithms. Fuzzy Sets and Systems, 5, 177–185, 1981.

[101] Wu, K., Yang, M: A cluster validity index for fuzzy clustering. Pattern Recognition Letters, 26, 1275–1291, 2005.

[102] Zhang, J-S., Leung, Y-W.: Improved possibilistic c-means clustering algorithms. IEEE Transactions on Fuzzy Systems, 12(2), 209–217, 2004.

[103] Zhang, L., Zhang, A., Ramanathan, M.: VizStruct: exploratory visualization for gene expression profiling. Bioinformatics, 20(1), 85–92, 2004.

Index