Hierarchical Subspace Clustering

Dissertation im Fach Informatik
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

von

Elke Achtert

Tag der Einreichung: 19.03.2007
Tag der mündlichen Prüfung: 24.04.2007

Berichterstatter:
Prof. Dr. Christian Böhm, Ludwig-Maximilians-Universität München
Prof. Dr. Jörg Sander, University of Alberta (Kanada)

Acknowledgement

A lot of people supported and encouraged me while I was working at my thesis. I am very grateful for all the help I got during this time. Unfortunately, I can only mention some of them here, but my thanks goes, of course, to all of them.

First of all, I want to express my dearest thanks to my supervisor and first referee on this thesis, Prof. Dr. Christian Böhm. He initiated and supported this work with his great experiences and the organizational background and gave me the opportunity to work on this challenging domain. I am also very grateful to Prof. Dr. Hans-Peter Kriegel who created a great working atmosphere within the database research group. Furthermore, I warmly thank Prof. Dr. Jörg Sander for his interest in my work and his immediate willingness to act as the second referee on this thesis.

This work was inspired by many discussions and cooperations with my colleagues. Without them this work could never have grown. I am very grateful for all the support I got during the past years and of course, not to forget, for all the fun we had. In particular, I want to thank Arthur Zimek, Dr. Peer Kröger, Dr. Alexey Pryakhin, Dr. Matthias Schubert, and Karsten Borgwardt for constructive and productive team-work, as well as Dr. Matthias Renz, Dr. Stefan Brecheisen, Johannes Aßfalg and Christian Marth for many helpful discussions. Special thanks to Dr. "Pete" Peter Kunath, who was always on the spot when I had trouble with technical and any other things.

The background help of Susanne Grienberger, who managed much of the administrative burden was another reason that working in this group was that comfortable. She read this thesis carefully and gave me invaluable hints for improving the language.

Furthermore, I want to express special thanks to Franz Krojer, who helped to master all technical issues. He promptly provided tools that helped to accelerate my work.

I also appreciate the substantial help of the students whose study thesis or diploma thesis I supervised. They helped me to manage the large amount of necessary tasks including implementation, data processing, and testing.

Last but not least, I like to express my deepest thanks to my family and my friends for their support and encouragement during the time that I was engaged in this study. But especially I want to thank my husband Frank for his patience enduring my absent-mindedness and for many sacrifices he shouldered in the last months. He always believed in me, sometimes more than I did.

Elke Achtert

Munich, March 2007

Abstract

It is well-known that traditional clustering methods considering all dimensions of the feature space usually fail in terms of efficiency and effectivity when applied to high-dimensional data. This poor behavior is based on the fact that clusters may not be found in the high-dimensional feature space, although clusters exist in subspaces of the feature space. To overcome these limitations of traditional clustering methods, several methods for subspace clustering have been proposed recently. Subspace clustering algorithms aim at automatically identifying lower dimensional subspaces of the feature space in which clusters exist.

There exist two types of subspace clustering algorithms: Algorithms for detecting clusters in axis-parallel subspaces and, as an extension, algorithms for finding clusters in subspaces which are arbitrarily oriented. Generally, the subspace clusters may be hierarchically nested, i.e., several subspace clusters of low dimensionality may form a subspace cluster of higher dimensionality. Since existing subspace clustering methods are not able to detect these complex structures, hierarchical approaches for subspace clustering have to be applied.

The goal of this dissertation is to develop new efficient and effective methods for hierarchical subspace clustering by identifying novel challenges for the hierarchical approach and proposing innovative and solid solutions for these challenges.

The first Part of this work deals with the analysis of hierarchical subspace clusters in axis-parallel subspaces. Two new methods are proposed that search simultaneously for subspace clusters of arbitrary dimensionality in order to detect complex hierarchies of subspace clusters. Furthermore, a new visualization model of the clustering result by means of a graph representation is provided.

In the second Part of this work new methods for hierarchical clustering in arbitrarily oriented subspaces of the feature space are discussed. The so-called correlation clustering can be seen as an extension of axis-parallel subspace clustering. Correlation clustering aims at grouping the data set into subsets, the so-called correlation clusters, such that the objects in the same correlation cluster show uniform attribute correlations. Two new hierarchical approaches are proposed which combine density-based clustering with Principal Component Analysis in order to identify hierarchies of correlation clusters.

The last Part of this work addresses the analysis and interpretation of the results obtained from correlation clustering algorithms. A general method is introduced to extract quantitative information on the linear dependencies between the objects of given correlation clusters. Furthermore, these quantitative models can be used to predict the probability that an object is created by one of these models.

Both, the efficiency and the effectiveness of the presented techniques are thoroughly analyzed. The benefits over traditional approaches are shown by evaluating the new methods on synthetic as well as real-world test data sets.

Abstract (in German)

Beim Clustering hochdimensionaler Daten erweisen sich oftmals traditionelle Clusteringverfahren, die sämtliche Dimensionen des Datenraums berücksichtigen, als ineffizient und ineffektiv. Dies ergibt sich aus der Tatsache, dass im hochdimensionalen Gesamtdatenraum möglicherweise keine Cluster entdeckt werden können, obwohl die Daten in Unterräumen des Datenraums Cluster bilden. Zur Beseitigung der Nachteile traditioneller Clusteringverfahren wurden eine Vielzahl sogenannter Subspace-Clustering-Algorithmen entwickelt, deren Ziel es ist, Cluster zu identifizieren, die in niedriger dimensionalen Unterräumen (Subspaces) des Gesamtdatenraums existieren.

Man unterscheidet zwei Arten von Subspace-Clustering-Algorithmen: Algorithmen zur Suche von Clustern in achsenparallelen Unterräumen des Gesamtdatenraums sowie, als Erweiterung, Algorithmen, die auf beliebig orientierten Unterräumen angewendet werden können. Die Cluster in den einzelnen Unterräumen können dabei auch ineinander geschachtelt sein, d.h. ein höher dimensionaler Unterraum kann Cluster enthalten, die ihrerseits Cluster in niedriger dimensionalen Unterräumen bilden. Da existierende Subspace-Clustering-Verfahren nicht in der Lage sind derartige Strukturen zu entdecken, sind hierzu neuartige hierarchische Ansätze für das Subspace-Clustering notwendig.

Das Ziel dieser Doktorarbeit ist es, effektive und effiziente Algorithmen im Bereich des hierarchischen Subspace-Clusterings zu entwickeln. Es werden neue Anwendungs- und Problemfelder für den hierarchischen Ansatz erschlossen und Lösungen für die resultierenden Probleme vorgestellt.

Der erste Teil der Arbeit beschäftigt sich mit der Analyse von hierarchischen Subspace-Clustern in achsenparallelen Unterräumen. Dabei werden zwei neuartige Verfahren vorgestellt, die simultan nach Subspace-Clustern unterschiedlicher Dimensionalität suchen, um komplexe Hierarchien von Subspace-Clustern aufzudecken. Des Weiteren wird ausführlich auf eine geeignete Visualisierung des Clustering-Ergebnisses mittels Graphen eingegangen.

Im zweiten Abschnitt der Arbeit werden neue Methoden zur hierarchischen Clusteranalyse in beliebig orientierten Unterräumen des Gesamtdatenraums behandelt. Das sogenannte Correlation-Clustering kann als Erweiterung des achsenparallelen Subspace-Clusterings betrachtet werden, um Objektgruppen, sogenannte Correlation-Cluster, in einer Datenbank zu ermitteln, die einheitliche Attribut-Korrelationen aufweisen. Dazu werden zwei hierarchische Ansätze beschrieben, die dichtebasiertes Clustering mit Hauptachsentransformation verbinden, um Hierarchien von Correlation-Clustern zu identifizieren.

Der letzte Teil der Arbeit erstreckt sich auf die Analyse und Interpretation der Ergebnisse von Clustering-Algorithmen für Unterräume. Dazu wird ein Verfahren vorgestellt, dass für gegebene Correlation-Cluster jeweils ein Modell zur Interpretation der linearen Abhängigkeiten der Objekte innerhalb des Clusters ableitet. Darüber hinaus können diese Modelle benutzt werden, um die Wahrscheinlichkeit zu ermitteln, dass ein Objekt von einem dieser Modelle erzeugt wurde.

Die Effizienz und Effektivität der vorgestellten Techniken wird detailliert untersucht und die Vorteile gegenüber herkömmlichen Verfahren sowohl mittels synthetischen Daten als auch realen Daten experimentell nachgewiesen.

Survey of Chapters

Contents

Part I

Preliminaries

Chapter 1

Introduction

The amount of data being collected in databases today far exceeds the ability to reduce and analyze data without the use of automated analysis techniques. Knowledge Discovery in Databases (KDD) is an interdisciplinary field that is evolving to provide automated analysis solutions. The core part of the KDD process is the application of specific data mining methods for pattern discovery and extraction. Section 1.1 introduces first the main concepts of Knowledge Discovery in Databases. Afterwards the data mining step is described in more detail and the most prominent methods on data mining are reviewed. As this thesis focus on hierarchical subspace clustering, a brief motivation for the special requirements when clustering high-dimensional data, leading to the paradigm of subspace clustering is given in Section 1.2. Section 1.3 concludes this Chapter with an outline of this thesis.

1.1 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [FPSS96]. The KDD process consists of an iterative sequence of the following steps (see Figure 1.1 for an illustration):

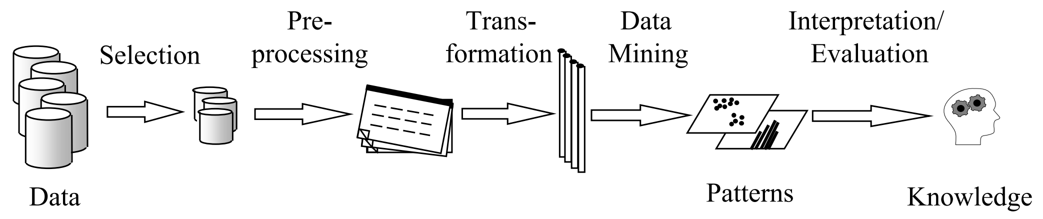


Figure 1.1: The KDD process.

- **Selection:** Creating a target data set by selecting a data set or focusing on a subset of attributes or data samples.
- **Preprocessing:** Performing data cleaning operations, such as removing noise, handling missing data fields, accounting for time-sequence information, etc.
- **Transformation:** Finding useful features to represent the data, e.g., using dimensionality reduction or transformation methods to reduce the number of attributes or to find invariant representations for the data.
- **Data Mining:** Searching for patterns of interest in a particular representation form, e.g., by applying classification rules, regression analysis, or clustering algorithms to the transformed data.
- **Interpretation and Evaluation:** Applying visualization and knowledge representation techniques to the extracted patterns. The user may return to previous steps of the KDD process if the results are unsatisfactory.

Since data mining is the core step of the KDD process, the notions “KDD” and “Data Mining” are often used as synonyms. In [FPSS96] data mining is defined as a step in the KDD process which consists of applying data analysis algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data. Existing data mining algorithms can be classified according to the following data mining methods [HK01]:

- **Characterization and Discrimination:** Summarization and comparison of general features of objects.
- **Association Analysis:** Discovering association rules showing attribute value conditions that occur frequently together in a given data set.
- **Classification and Prediction:** Supervised learning of models or functions to organize (new) data objects into predefined classes.
- **Evolution Analysis:** Modeling trends in time related data that change in time.
- **Clustering:** Unsupervised grouping of the data objects into classes by maximizing the similarity between objects of the same class and minimizing the similarity between objects of different classes.
- **Outlier Analysis:** Identifying data objects that cannot be grouped in a given class or cluster, since they do not correspond to the general model of the data.

1.2 Subspace Clustering

The general topic of this thesis is clustering, one of the primary data mining tasks. All clustering algorithms aim at segmenting a collection of objects into subsets or "clusters", such that those objects within one cluster are more closely related to one another than to objects assigned to different clusters. Thus, clustering groups the objects of a data set into clusters by maximizing the intra-cluster similarity and minimizing the inter-cluster similarity. Finding these clusters is important because they represent different classes of objects that were previously unknown, bringing additional insight which can be exploited for various purposes.

Many applications of clustering are characterized by high-dimensional data which poses two challenges: First, clusters may be only visible in certain, maybe even arbitrarily oriented subspaces of the feature space. The second challenge is the so-called curse of dimensionality which essentially means

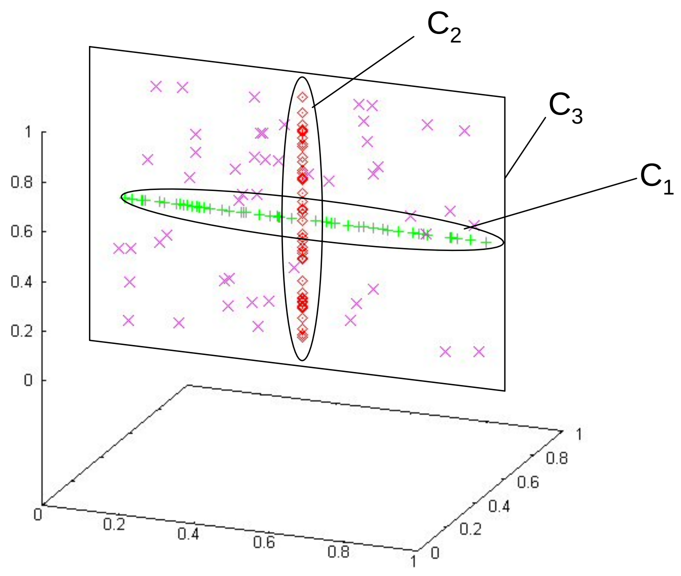


Figure 1.2: Hierarchy of subspace clusters.

that distance measures become increasingly meaningless as the number of dimensions increases in the data set. Thus, traditional clustering methods considering all dimensions of the feature space often fail to detect meaningful clusters in high-dimensional data.

To overcome the limitations of traditional clustering methods, recent research has focused on discovering clusters embedded in different subspaces of high-dimensional data sets, a paradigm commonly known as subspace clustering. Subspace clustering algorithms can be classified in two groups:

1. Algorithms that aim at finding clusters in axis-parallel subspaces of the data space. These methods are also called *axis-parallel subspace clustering* or *projected clustering*.

2. Algorithms for finding clusters in arbitrarily oriented subspaces of the feature space. These methods are also called *oriented clustering*, *generalized subspace clustering*, or *correlation clustering*.

Generally, the subspace clusters may be hierarchically nested, i.e., several subspace clusters of low dimensionality may together form a subspace cluster of higher dimensionality. Figure 1.2 illustrates a simple example of a hierarchy of subspace clusters in a 3-dimensional feature space: the clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ are embedded within cluster $\mathcal{C}_3$. Since existing subspace clustering methods are not able to detect these complex structures, hierarchical approaches for subspace clustering have to be applied. In this thesis, the research field of hierarchical subspace clustering is addressed and innovative and solid solutions for identifying hierarchically nested clusters that only appear in subspaces of the entire feature space are proposed.

1.3 Outline of the Thesis

The remainder of this thesis is organized as follows:

Part I deals with the preliminaries.

Chapter 1 should give the reader a short introduction to the broader context of this thesis.

Chapter 2 provides a brief and rather general overview of traditional hierarchical clustering methods.

Chapter 3 introduces first the density-based notion of clusters underlying the algorithm DBSCAN. Then, its hierarchical extension, leading to the notion of hierarchical density-based clustering which constitutes the central concept of the algorithm OPTICS is discussed in detail.

Part II deals with the analysis of hierarchical subspace clusters in axis-parallel subspaces. Two new methods are proposed that search simultaneously for subspace clusters of arbitrary dimensionality in order to detect complex hierarchies of subspace clusters.

Chapter 4 gives an introduction and motivation to the research area of hierarchical axis-parallel subspace clustering.

Chapter 5 reviews current approaches on axis-parallel subspace clustering.

Chapter 6 proposes the novel algorithm HiSC (Hierarchical Subspace Clustering) which is the first subspace clustering algorithm for detecting hierarchies of subspace clusters. HiSC can detect clusters in subspaces of significantly different dimensionality and is able to determine hierarchies of nested subspace clusters. Several comparative experiments show the superiority of HiSC to existing methods.

Chapter 7 presents a major extension of HiSC called DiSH (Detecting Subspace Cluster Hierarchies). While HiSC is limited to single inclusions of subspace cluster hierarchies, DiSH is able to determine hierarchies of single and multiple inclusions. Furthermore, DiSH computes a clear and intuitive graph representation of the result such that the complete relationships among subspace clusters can be seen at a glance. A broad experimental evaluation shows the superior accuracy of DiSH compared to its competitors.

Part III discusses two new methods for hierarchical clustering in arbitrarily oriented subspaces of the feature space. The so-called correlation clustering can be seen as an extension of axis-parallel subspace clustering. Correlation clustering aims at grouping the data set into subsets, the so-called correlation clusters, such that the objects in the same correlation cluster show uniform attribute correlations.

Chapter 8 gives an introduction and motivation to the research area of hierarchical subspace clustering in arbitrarily oriented subspaces of the feature space.

Chapter 9 provides related work on subspace clustering algorithms for arbitrarily oriented subspaces.

Chapter 10 proposes the new algorithm HiCO (Hierarchical Correlation Ordering), the first algorithm for computing hierarchies of correlation clusters. In contrast to existing approaches, HiCO does not require the user to specify any global density threshold, the number of clusters to be found, nor any parameter specifying the dimensionality of the correlations. The extensive experimental evaluation shows that HiCO finds meaningful and rich hierarchies of correlation clusters in synthetic and real-world data sets.

Chapter 11 introduces the novel clustering algorithm ERiC (Exploring complex hierarchical Relationships among Correlation clusters) to efficiently detect hierarchical relationships between correlation clusters also allowing multiple inclusions. The resulting cluster hierarchy is visualized by means of a graph model. A broad experimental evaluation shows that ERiC finds more information than state-of-the art correlation clustering methods and outperforms existing competitors in terms of efficiency.

Part IV addresses the analysis and interpretation of the results obtained from correlation clustering algorithms. An original approach to derive quantitative information on the linear dependencies within correlation clusters is proposed. The concepts are independent of the clustering model and thus, can be applied as a post-processing step to any correlation clustering algorithm.

Chapter 12 gives a short introduction to the challenge of advanced data analysis and system modeling.

Chapter 13 reviews related work on existing approaches for deriving descriptions of quantitative dependencies among several attributes.

Chapter 14 formalizes the notion of PCA-based correlation clusters.

Chapter 15 proposes the concepts to derive quantitative models of correlation clusters. Furthermore, it is sketched how these quantitative models can be used to predict the probability that an object is created by one of these models.

Chapter 16 presents a broad experimental evaluation where the practical importance of the new approach is demonstrated.

Part V concludes this thesis.

Chapter 17 summarizes and discusses the major contributions of this thesis. It concludes with pointing out some future directions.

Chapter 2

Hierarchical Clustering

In the past decade, many algorithms for the problem of clustering have been proposed. There are two major methods of clustering – *partitioning clustering* and *hierarchical clustering*. Partitioning clustering methods compute a "flat" partition of the data set, i.e., they produce a unique assignment of each data object to a cluster. The number of clusters k is often a user specific parameter. Prominent example algorithms for such partitioning clustering are k-means [McQ67], PAM [KR90], CLARANS [NH94], and the EM-algorithm [DLR77]. Density-based methods like the DBSCAN-algorithm [EKSX96] also assign each object to a unique cluster or noise. Since this clustering notion is basis of this thesis, the concepts underlying DBSCAN are presented in detail in Section 3.1.

In hierarchical clustering, the data are not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place which may run from a single cluster containing all objects to n clusters each containing a single object. As this thesis copes with the challenge of extending traditional hierarchical clustering to hierarchical subspace clustering, in the following Sections a brief and rather general overview of traditional hierarchical clustering methods is given. The hierarchical clustering algorithm OPTICS [ABKS99] combines the notion of density-based clustering and hierarchical clustering. Since the enhancements to hierarchical subspace clustering presented in Part II and III are based to some extent on the concepts of this

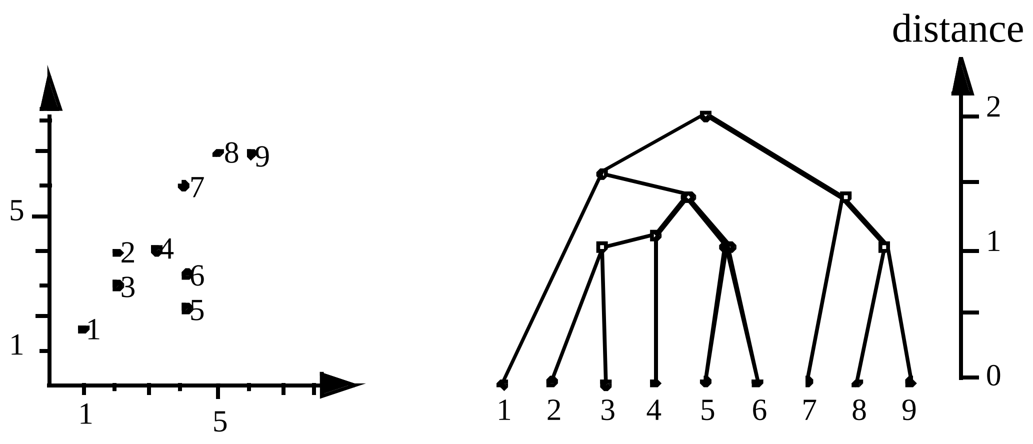


Figure 2.1: A dendrogram (right) for a sample data set (left).

algorithm, OPTICS are discussed in more detail in section 3.2.

2.1 Concepts of Hierarchical Clustering

Hierarchical clustering algorithms compute a hierarchical decomposition of the data objects instead of a unique assignment of data objects to clusters. Given the data set $\mathcal{D}$, the goal is to produce a 2-dimensional diagram known as *dendrogram* in which nodes represent subsets of $\mathcal{D}$, simulating the structure found in $\mathcal{D}$ with the following properties (cf. Figure 2.1):

- The root of the dendrogram is the whole data set $\mathcal{D}$.
- Each leaf of the dendrogram corresponds to one data object of $\mathcal{D}$.
- The internal nodes of the dendrogram are defined as the union of their children, i.e., each node represents a cluster containing all objects of the leaf nodes below this node.
- Each level of the dendrogram represents a partition of the data set into several nested clusters.

Hierarchical Clustering is subdivided into *agglomerative* methods which proceed by series of fusions of data objects into clusters, and *divisive* methods which separate the data objects successively into finer clusters. Agglomerative hierarchical clustering algorithms follow a bottom-up strategy by merging the clusters iteratively. They start by placing each data object in its own single cluster. Then, the clusters are merged together into larger and larger clusters by grouping similar data objects together until the entire data set is encapsulated into one final root cluster. Most hierarchical methods belong to this category. They differ only in their definition of between-cluster similarity.

Divisive hierarchical clustering works the opposite way around - it starts with all data objects in one root cluster and subdivides them into smaller clusters until each cluster consists of only one single data object. Divisive methods are not generally available, and rarely have been applied. The reasons for this is mainly computational - divisive clustering is more computationally expensive when deciding to divide one cluster in two, given all possible choices. While in agglomerative procedures in one step two out of maximum n elements have to be chosen for merging, in divisive procedures fundamentally all subsets have to be analyzed so that divisive procedures have an algorithmic complexity of $O(2^n)$.

Agglomerative procedures have the drawback that an incorrect merging of clusters in an early stage often yields results which are far away from the real cluster structure. Divisive procedures immediately start with interesting cluster arrangements and are therefore more robust. Since usually agglomerative procedures are used because of their efficiency, the agglomerative algorithm will be explained in detail in the following.

2.2 Basic Algorithm for Agglomerative Hierarchical Clustering

Given a data set $\mathcal{D}$ of n objects to be clustered, the basic process of agglomerative hierarchical clustering is defined as follows:

1. Place each data object $o_i \in \mathcal{D}$ $(i = 1, \dots n)$ in its own single cluster $\mathcal{C}_i$.

Create the list of initial clusters $\mathbf{C} = \mathcal{C}_1, \ldots, \mathcal{C}_n$ which will build the leaves of the resulting dendrogram.

2. Find the two clusters $\mathcal{C}_i, \mathcal{C}_j \in \mathbf{C}$ with the minimum distance to each other.

3. Merge the clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ to create a new internal node $\mathcal{C}_{ij}$ which will be the parent of $\mathcal{C}_i$ and $\mathcal{C}_j$ in the resulting dendrogram. Remove $\mathcal{C}_i$ and $\mathcal{C}_j$ from $\mathbf{C}$.

4. Repeat step 2 and 3 until the total number of clusters in $\mathbf{C}$ becomes one.

In the first step of the algorithm, when each object represents its own cluster, the distances between the clusters are defined by the chosen distance function between the objects of the data set. However, once several objects have been linked together, a linkage rule is needed to determine the actual distance between two clusters. There are numerous linkage rules that have been proposed. In the following Section some of the most commonly used linkage methods are presented.

2.3 Linkage Methods

Let $\mathcal{D}$ be a data set, $\textsc{Dist}$ denote the distance function between the data objects of $\mathcal{D}$, and $\mathcal{C}_i$ and $\mathcal{C}_j$ be two disjunct clusters consisting of objects of $\mathcal{D}$, i.e., $\mathcal{C}_i, \mathcal{C}_j \subseteq \mathcal{D}$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$. In the following, some of the most commonly used linkage rules to determine the distance between two clusters are described.

Single-Link. One of the simplest agglomerative hierarchical clustering methods is the Single-Link method [Sib73], also known as the nearest neighbor technique. Single-Link defines the distance $\textsc{Dist}_{SL}$ between any two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ as the minimum distance between them, i.e., as the distance between the two closest objects:

$$\textsc{Dist}_{SL}(\mathcal{C}_i, \mathcal{C}_j) = \min_{x_i \in \mathcal{C}_i, x_j \in \mathcal{C}_j} \{\textsc{Dist}(x_i, y_i)\}.$$

Using the Single-Link method often causes the *chaining phenomenon*, also called *Single-Link effect*, which is a direct consequence of the Single-Link approach tending to force clusters together due to single objects being close to each other regardless of the positions of other entities in that cluster.

Complete-Link. The Complete-Link method [Def77], also called farthest neighbor technique, is the opposite of Single-Link. Complete-Link defines the distance $\textsc{Dist}_{CL}$ between any two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ as the maximum distance between them:

$$\textsc{Dist}_{CL}(\mathcal{C}_i, \mathcal{C}_j) = \max_{x_i \in \mathcal{C}_i, x_j \in \mathcal{C}_j} \{\textsc{Dist}(x_i, y_i)\}.$$

The Complete-Link method should not be used if there is a lot of noise expected to be present in the data set. It also produces very compact clusters. This method is useful if one is expecting objects of the same cluster to be far apart in multidimensional space (provided there is no noise). In other words, outliers are given more weight in the cluster decision.

Average-Link. Average-Link [Voo86] takes the mean distance between all possible pairs of objects belonging to the two clusters in question. Therefore, it is more computationally expensive to compute the distance $\textsc{Dist}_{AVG}$ between any two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ than the before mentioned methods:

$$\textsc{Dist}_{AVG}(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i| \cdot |\mathcal{C}_j|} \sum_{x_i \in \mathcal{C}_i, x_j \in \mathcal{C}_j} \{\textsc{Dist}(x_i, y_i)\}.$$

Average-Link is sometimes also called UPGMA (Unweighted Pair-Group Method using Arithmetic averages). There are several other variations of this method, e.g. Weighted Pair-Group Average, Unweighted Pair-Group Centroid, Weighted Pair-Group Centroid, but one should understand that it is a halfway-house between Single-Link and Complete-Link. The chaining problem is not observed for this method and outliers are not given any special favor in the cluster decision, which makes this method very popular.

Wards method This method is distinct from all other methods because it uses an analysis of variance approach to evaluate the distances between clusters. In short, this method attempts to minimize the sum of squares of any two (hypothetical) clusters that can be formed at each step. Generally, this method is regarded as very efficient, however, it tends to create clusters of small size.

Chapter 3

Density-Based Clustering

Many clustering algorithms have been proposed in recent years. This thesis is especially based on the density-based clustering approach which turned out to be one of the most effective and also efficient ones. The key concept of density-based clustering is the observation that inside a cluster the density of points is considerably higher than outside a cluster. Furthermore, different clusters are separated by areas of noise, where the density is lower than inside a cluster.

In this Chapter, an introduction to the density-based notion of clusters is given. In particular, first, in Section 3.1 the notion of density-connectivity underlying the algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [EKSX96] is introduced. Then, in Section 3.2 its hierarchical extension leading to the notion of hierarchical density-based clustering as proposed in [ABKS99] which constitutes the central concept of the algorithm OPTICS (Ordering Points To Identify the Clustering Structure) is discussed.

3.1 Foundations of Density-Based Clustering

The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius ε has to contain at least a specified minimum number μ of objects, i.e., the cardinality of the neighborhood has to exceed a given threshold. In the following, the basic definitions of density-based clustering are presented.

Let $\mathcal{D}$ be a data set of objects. The distance function between the objects of $\mathcal{D}$ is denoted by $\textsc{Dist}$. For any $\varepsilon \in \mathbb{R}^+$ the ε-neighborhood of an object $p \in DB$ is denoted by $\mathrm{N}_\varepsilon(p)$. More formally:

$$\mathrm{N}_\varepsilon(p) = \{o \in \mathcal{D} | \textsc{Dist}(p, o) \leq \varepsilon\}.$$

Definition 3.1 (direct density-reachability).
Let $\varepsilon \in \mathbb{R}^+, \mu \in \mathbb{N}^+, \mu \leq |\mathcal{D}|$*. An object* $p \in DB$ *is* directly density-reachable from object $q \in \mathcal{D}$ w.r.t. ε and μ *if* $|N_\varepsilon(q)| \geq \mu \wedge p \in N_\varepsilon(q)$*.*

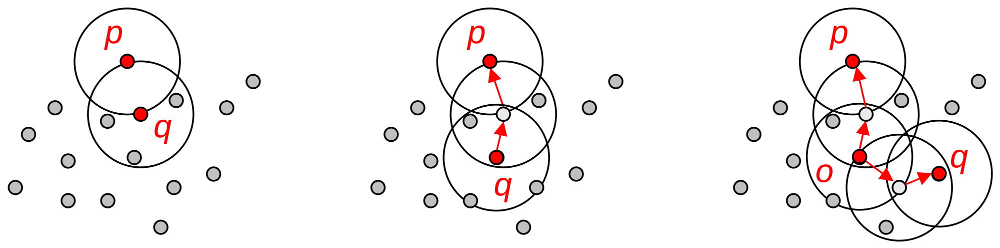
If the first condition $|\mathrm{N}_\varepsilon(q)| \geq \mu$ holds for an object q, q is called a *core point.*

Definition 3.2 (density-reachability).
Let $\varepsilon \in \mathbb{R}^+, \mu \in \mathbb{N}^+, \mu \leq |\mathcal{D}|$*. An object* $p \in DB$ *is* density-reachable from an object $q \in \mathcal{D}$ w.r.t. ε and μ *if there is a sequence of objects* $p_1, \ldots, p_n, p_1 = q, p_n = p$ *such that* p_{i+1} *is directly density-reachable from* p_i*.*

Definition 3.3 (density-connectivity).
Let $\varepsilon \in \mathbb{R}^+, \mu \in \mathbb{N}^+, \mu \leq |\mathcal{D}|$*. An object* $p \in DB$ *is* density-connected to object $q \in \mathcal{D}$ w.r.t. ε and μ *if there is an object* $o \in \mathcal{D}$ *such that both,* p *and* q*, are density reachable from* o*.*

Figure 3.1 illustrates the concepts of density-based clustering given above. Intuitively, a density-based cluster is defined to be a set of density-connected objects which is maximal w.r.t density-reachability. The objects in $\mathcal{D}$ not belonging to any cluster are defined as noise. A cluster consists of core objects inside the cluster and so-called border objects located at the border of the cluster which do not fulfill the core object condition. These border objects are directly density-reachable from at least one core object of the cluster.

Using the previously described concepts, the algorithm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) as proposed in [EKSX96]



(a) p is directly density-reachable from q, but q is not directly density-reachable from p.

(b) p is density-reachable from q.

(c) p and q are density-connected.

Figure 3.1: Density-based clustering concepts ($\mu = 4$).

computes a flat density-based decomposition w.r.t. the user-specified parameters ε and μ. DBSCAN is able to detect arbitrarily shaped clusters by one single pass over the data. To do so, DBSCAN uses the fact, that a density-connected cluster can be detected by finding one of its core points p and computing all objects which are density-reachable from p (cf. Lemma 1 and 2 in [EKSX96]). The retrieval of density-reachable objects is performed by iteratively collecting directly density-reachable objects. DBSCAN checks the ε-neighborhood of each point p in the database. If $\mathrm{N}_\varepsilon(p)$ of an object p consists of at least μ objects, i.e., if p is a core object, a new cluster $\mathcal{C}$ containing all objects of $\mathrm{N}_\varepsilon(p)$ is created. Then, the ε-neighborhood of all points $q \in \mathcal{C}$ which have not yet been processed is checked. If object q is also a core object, the neighbors of q which are not already assigned to cluster $\mathcal{C}$ are added to $\mathcal{C}$ and their ε-neighborhood is checked in the next step. This procedure is repeated until no new point can be added to the current cluster $\mathcal{C}$. Then the algorithm continues with a point which has not yet been processed, trying to expand a new cluster.

3.2 Hierarchical Density-Based Clustering

The algorithm OPTICS (Ordering Points To Identify the Clustering Structure) as proposed in [ABKS99] extends the density-connected clustering notion of DBSCAN by hierarchical concepts. In contrast to DBSCAN, OPTICS does not assign cluster memberships but computes an ordering in which the objects are processed and additionally generates the information which would be used by an extended DBSCAN algorithm to assign cluster memberships. This information consists of two values for each object, the core-distance and the reachability-distance. In the following, the definitions underlying the algorithm OPTICS are shortly introduced.

Let $\mathcal{D}$ be a data set of objects. The distance function between the objects of $\mathcal{D}$ is denoted by $\textsc{Dist}$. For any $\varepsilon \in \mathbb{R}^+$ the ε-neighborhood of an object $p \in DB$ is denoted by $\mathrm{N}_\varepsilon(p)$.

Definition 3.4 (core-distance).
Let $\varepsilon \in \mathbb{R}^+, \mu \in \mathbb{N}^+, \mu \leq |\mathcal{D}|$ and $\mu - \textsc{Dist}(p)$ denote the distance from object $p \in \mathcal{D}$ to its μ-nearest neighbor in $\mathcal{D}$ w.r.t. distance function $\textsc{Dist}$*. The* core-distance of p w.r.t. ε and μ *is defined as*

$$\textsc{CoreDist}_{\varepsilon,\mu}(p) = \begin{cases} \infty & \textit{if } |N_\varepsilon(p)| < \mu \\ \mu - \textsc{Dist}(p) & \textit{otherwise} \end{cases}.$$

Definition 3.5 (reachability-distance).
Let $\varepsilon \in \mathbb{R}^+, \mu \in \mathbb{N}^+, \mu \leq |\mathcal{D}|$. The reachability-distance of an object $q \in \mathcal{D}$ w.r.t. ε and μ relative to an object $p \in \mathcal{D}$ *is defined as*

$$\textsc{ReachDist}_{\varepsilon,\mu}(p,q) = \max\{\textsc{CoreDist}_{\varepsilon,\mu}(p), \textsc{Dist}(p,q)\}.$$

Figure 3.2 illustrates both concepts: The reachability-distance of object p relative to object o equals the core-distance of o. The reachability distance of object q relative to o is equal to the distance between q and o.

The result of OPTICS is an ordering of the objects, the so-called *cluster ordering* of the data set w.r.t. the two input parameters ε and μ.

Definition 3.6 (cluster ordering).
Let $\varepsilon \in \mathbb{R}^+, \mu \in \mathbb{N}^+, \mu \leq |\mathcal{D}|$, and CO be a totally ordered permutation of the objects in $\mathcal{D}$. Each object $p \in \mathcal{D}$ has additional attributes p.$\textsc{Pos}$*, p.*$\textsc{Core}$*,*

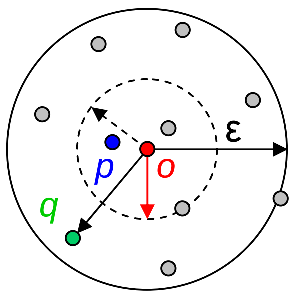

ε
p
o
q
core-distance of o
reachability-distance of p relative to o
reachability-distance of q relative to o

Figure 3.2: Illustration of core-distance and reachability distance for $\mu = 4$.

and $p.\textsc{Reach}$, *where* $p.\textsc{Pos} \in \{1, \dots, |CO|\}$ *symbolizes the position of* p *in* CO. CO *is called a cluster ordering w.r.t.* ε *and* μ *if the following conditions hold:*

1. $\forall p \in CO : p.\textsc{Core} = \textsc{CoreDist}_{\varepsilon,\mu}(p)$
2. $\forall p, q, r \in CO : p.\textsc{Pos} < q.\textsc{Pos} \wedge q.\textsc{Pos} < r.\textsc{Pos}$
   $\Rightarrow \textsc{ReachDist}_{\varepsilon,\mu}(p, q) \leq \textsc{ReachDist}_{\varepsilon,\mu}(p, r)$
3. $\forall p, q \in CO : q.\textsc{Reach} = \min\{\textsc{ReachDist}_{\varepsilon,\mu}(p, q) | p.\textsc{Pos} < q.\textsc{Pos}\}$,
   where $\min \emptyset = \infty$

Intuitively, condition 2 states that the cluster ordering is built on selecting at each position i in CO that object p having the minimum reachability-distance to any object before p in CO. The attribute $p.\textsc{Core}$ for any object $p \in \mathcal{D}$ denotes the core-distance of p, whereas the attribute $p.\textsc{Reach}$ is equal to the reachability-distance assigned to object p during the generation of the cluster ordering CO. The attribute $p.\textsc{Reach}$ is also called the reachability of object p.

The pseudocode of the algorithm OPTICS is depicted in Figures 3.3 and 3.4. It starts with an arbitrarily chosen object $p \in \mathcal{D}$, assigns a reachability of

```
algorithm OPTICS(Database $\mathcal{D}$, Real $\varepsilon$, Integer $\mu$)
    initialize empty cluster order $co$;
    initialize empty priority queue $pq$ ordered by $\textsc{Dist}$;

    for each $p \in \mathcal{D}$ do
        if $p \notin co$ do
            $p.\textsc{Reach} := \infty$;
            $p.\textsc{Core} := \textsc{CoreDist}_{\varepsilon,\mu}(p)$;
            $co$.add($p$);

            if $p.\textsc{Core} \neq \infty$ do
                update($pq, p, \varepsilon, co$);

                while $pq \neq \emptyset$ do
                    $q := pq$.next();
                    $q.\textsc{Core} := \textsc{CoreDist}_{\varepsilon,\mu}(q)$;
                    $co$.add($q$);

                    if $q.\textsc{Core} \neq \infty$ do
                        update($pq, q, \varepsilon, co$);
                    end if
                end while
            end if
        end if
    end for

    return $co$;

end.
```

Figure 3.3: The OPTICS algorithm.

∞ to object p and expands the cluster order if the core-distance of p is smaller than the specified ε-parameter. The expansion is worked out by inserting each object $q \in \mathrm{N}_\varepsilon(p)$ into a priority queue. The priority queue stores that object first, having the minimum reachability to all already processed objects. The heap structure is maintained by a procedure (cf. Figure 3.4) which updates the reachability of the objects that are already in the priority queue

```
procedure update(PriorityQueue $pq$, Object $p$, Real $\varepsilon$, ClusterOrder $co$)
    for each $q \in \mathbf{N}_\varepsilon(p)$ do
        if $q \notin co$ do
            $r := \max\{p.\textsc{Core}, \textsc{Dist}(p, q)\}$;
            if $q \in pq$ do
                if $q.\textsc{Reach} > r$ do
                    $q.\textsc{Reach} := r$;
                    $pq$.decrease($q$);
                end if
            else
                $q.\textsc{Reach} := r$;
                $pq$.add($q$);
            end if
        end if
    end for
end.
```

Figure 3.4: The procedure to update the priority queue.

if their according values decrease. The next object to be inserted in the cluster ordering is always the first object of the priority queue. If the core distance of this object is smaller or equal to ε, all points in the ε-neighborhood are again inserted into or updated in the priority queue. If the priority queue is empty and there are still some not yet processed points in $\mathcal{D}$, another not yet handled object in $\mathcal{D}$ is chosen to further expand the cluster ordering CO as described above.

The cluster structure can be visualized by so-called 2-dimensional *reachability plots* where the objects are plotted according to the sequence specified in the cluster ordering along the x-axis, and for each object, its reachability along the y-axis. Figure 3.5 (right) depicts the reachability plot based on the cluster ordering computed by OPTICS for the sample 2-dimensional data set in Figure 3.5 (left). Valleys in this plot indicate clusters: objects having a

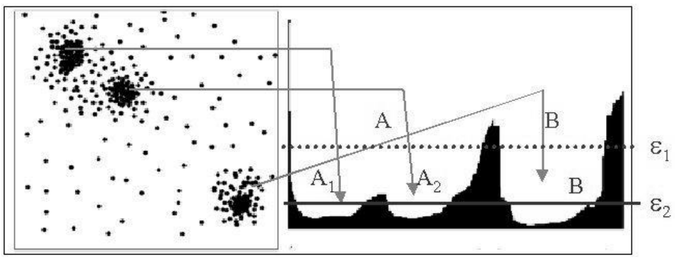


Figure 3.5: Reachability plot (right) computed by OPTICS for a sample 2-dimensional data set (left).

small reachability value are closer and thus more similar to their predecessor objects than objects having a higher reachability value.

The reachability plot generated by OPTICS can be cut at any level ε' parallel to the x-axis. It represents the density-based clusters according to the density threshold ε': A consecutive subsequence of objects having a smaller reachability value than ε' belongs to the same cluster. An example is presented in Figure 3.5: For a cut at the level ε_1, two clusters A and B can be found. Compared to this clustering, a cut at level ε_2 would yield three clusters. The cluster A is split into two smaller clusters denoted by A1 and A2 and cluster B decreased its size. This illustrates, how the hierarchical cluster structure of a database is revealed at a glance and can be easily explored by visual inspection.

Part II

Hierarchical Axis-Parallel Subspace Clustering

Chapter 4

Introduction

The well-known curse of dimensionality usually limits the applicability of traditional clustering algorithms to high-dimensional feature spaces because different sets of features are relevant for different (subspace) clusters. To detect such lower dimensional axis-parallel subspace clusters, the task of subspace clustering (or projected clustering) has been defined recently. Subspace clustering can be seen as an extension of traditional clustering which aims at automatically identifying lower dimensional axis-parallel subspaces of the feature space in which clusters exist. For the sake of brevity, an axis-parallel subspace cluster is shortly called subspace cluster in the following. A subspace cluster that is associated to a λ-dimensional projection/subspace, i.e., that it is spanned by λ attributes, is called a *λ-dimensional subspace cluster*. The dimensionality of a subspace associated to a subspace cluster is called *subspace dimensionality*.

Subspace clustering algorithms can be classified by the type of results they produce. The first class of algorithms allows overlapping clusters, i.e., one point may belong to different clusters in different projections. The second class of subspace clustering algorithms generates non-overlapping clusters and assigns each object to a unique cluster or noise. Please refer to Chapter 5 for a detailed discussion on both types of existing subspace clustering approaches. Algorithms that allow an overlap usually produce a vast amount of clusters which is hard to interpret. Thus, in the following discussion only

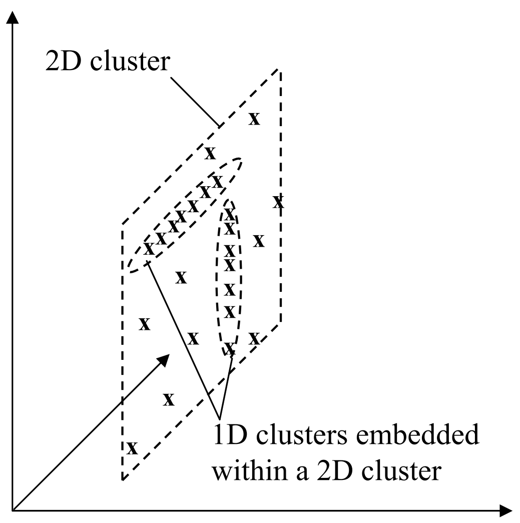


Figure 4.1: Hierarchically nested subspace clusters.

algorithms that generate non-overlapping clusters are considered.

The existing algorithms for non-overlapping subspace clustering - as mentioned in Chapter 5 - usually have one severe limitation in common. These algorithms will miss important information about the clustering structure in case of hierarchically nested subspace clusters, i.e., if several subspace clusters of low dimensionality may together form a larger subspace cluster of higher dimensionality. For example, consider two axis-parallel lines in a 3-dimensional space that are embedded into an axis-parallel 2-dimensional plane (cf. Figure 4.1). Each of the two lines forms a 1-dimensional subspace cluster. On the other hand, the plane is a 2-dimensional subspace cluster that includes the two 1-dimensional subspace clusters. In order to detect the lines, a search for 1-dimensional subspace clusters has to be applied, whereas in order to detect the plane, a search for 2-dimensional subspace clusters has to be performed. Moreover, searching subspace clusters of different dimensionality is basically a hierarchical problem, because the information that a point belongs to some k-dimensional subspace cluster that is itself embedded into an l-dimensional subspace cluster ($k < l$) can only be uncovered by using a hierarchical approach. None of the previously proposed algorithms for subspace clustering is able to detect such hierarchies of nested subspace clusters.

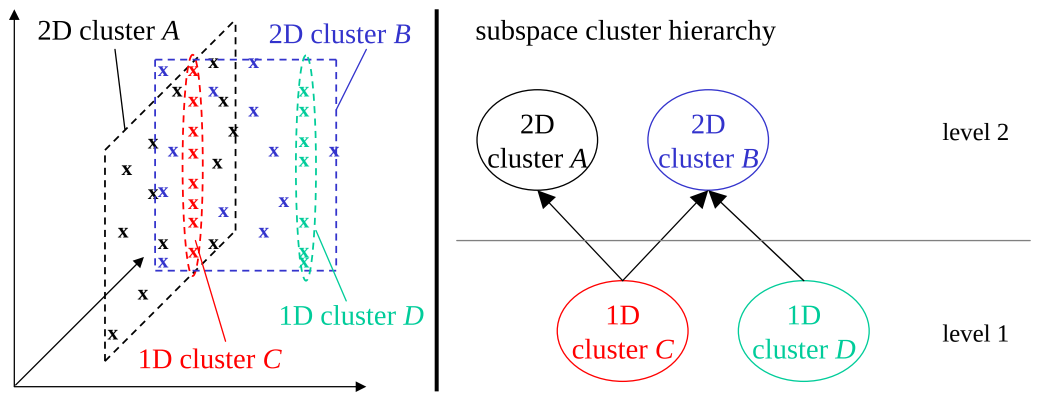


Figure 4.2: Hierarchies of subspace clusters with multiple inclusion.

A related problem of many existing approaches is that the dimensionality of the subspace clusters needs to be specified in advance. However, if the dimensionalities of the subspace clusters vary significantly, these methods will most likely miss important clusters.

A second limitation of some approaches for non-overlapping subspace clustering is the use of a clustering model that relies on a global density threshold. The usage of a global density threshold that all clusters must exceed is rather inapplicable since higher dimensional subspace clusters will most likely be less dense than lower dimensional subspace clusters. In addition, different clusters in one single subspace may exhibit significantly different densities. To discover clusters of different densities, a hierarchical approach has to be applied.

A third limitation derives from the fact that subspace clusters may be hierarchically nested exhibiting multiple inclusions. For instance, a subspace cluster of low dimensionality may be embedded within *several* larger subspace clusters of higher dimensionality. None of the existing algorithms is able to detect such important complex hierarchical relationships among the subspace clusters. An example of such a hierarchy is depicted in Figure 4.2 (left). Two 1-dimensional (1D) cluster (C and D) are embedded within one 2-dimensional (2D) cluster (B). In addition, cluster C is embedded within both 2-dimensional clusters A and B. Detecting such relationships of subspace clusters is obviously a hierarchical problem. The resulting hierarchy is different from the result of a conventional hierarchical clustering algorithm, e.g., a dendrogram. In a dendrogram, each object is placed in a singleton cluster at the leaf level, whereas the root node represents the cluster consisting of the entire data set. Any inner node n represents the cluster consisting of the points located in the subtree of n. Dendrograms are limited to single inclusion, i.e., a lower dimensional cluster can only be the child cluster of one higher dimensional cluster. However, hierarchies of subspace clusters may exhibit multiple inclusions, e.g., cluster C in Figure 4.2 is a child of cluster A and B. The concept of multiple inclusions is similar to that of "multiple inheritance" in software engineering. To visualize such more complex relationships among subspace clusters, graph representations are needed rather than tree representations. An appropriate visualization model to visualize the complex hierarchies, the so-called *subspace clustering graph*, will be proposed in Chapter 7. The subspace clustering graph consists of nodes at different levels (cf. Figure 4.2 (right)). These levels represent the dimensionality λ of the subspace in which the cluster is found. For instance, the level of subspace cluster A in the graph of Figure 4.2 is two, because A forms a 2-dimensional subspace cluster. Each object p is assigned to a unique node in the graph, representing the lowest dimensional subspace cluster in which p is placed. In addition, an edge between a k-dimensional cluster C and an l-dimensional cluster B, where $l > k$, such as in Figure 4.2, indicates that all points of cluster C are also members of cluster B.

In this Part, two new hierarchical approaches for subspace clustering are proposed. First, Chapter 5 surveys related work in the area of axis-parallel subspace clustering. Then, Chapter 6 introduces the algorithm HiSC (Hierarchical Subspace Clustering) which overcomes the first two limitations of the state-of-the-art subspace clustering approaches mentioned above. HiSC is a new algorithm that applies a hierarchical approach to subspace clustering and searches simultaneously for subspace clusters of arbitrary dimensionality detecting hierarchies of subspace clusters. HiSC follows the Single-Link approach and does not require the user to specify critical parameters such as a global density threshold for the points within a cluster or the dimensionality of the subspace clusters which has been limiting the quality of several previously proposed subspace clustering algorithms.

In Chapter 7 the algorithm DiSH (Detecting Subspace cluster Hierarchies) is proposed which includes all advantages of HiSC, and furthermore, can cope adequately with the third restriction of existing subspace clustering approaches mentioned before. DiSH improves in the following aspects over the state-of-the-art subspace clustering approaches: First, DiSH uncovers complex hierarchies of nested subspace clusters including multiple inclusions. Second, DiSH can detect clusters in subspaces of significant different dimensionality. Third, DiSH is able to detect clusters of different size, shape, and density. Besides, a new visualization method, the so-called subspace clustering graph is proposed to visualize the resulting complex hierarchies by means of an appropriate visualization model. Utilizing this visualization method, the relationships between the subspace clusters can be explored at a glance.

Chapter 5

Related Work

Subspace clustering algorithms can be distinguished by the type of results they produce. One class of algorithms aims at finding all clusters in all subspaces of the feature space producing overlapping clusters, i.e., one point may belong to different subspace clusters in different subspaces. Prominent examples of such algorithms include e.g. CLIQUE [AGGR98], ENCLUS [CFZ99], MAFIA [GNC99], SUBCLU [KKK04], and FIRES [KKRW05]. These algorithms are discussed in Section 5.1. The second class of subspace clustering algorithms, such as PROCLUS [APW^{+}99], DOC [PJAM02], and PreDeCon [BKKK04], focus on finding non-overlapping subspace clusters. These methods assign each point to a unique subspace cluster or noise and are reviewed in Section 5.2. Please note that none of the existing approaches to axis-parallel subspace clustering can detect hierarchies of nested subspace clusters.

5.1 Overlapping Algorithms

CLIQUE (CLustering In QUEst) [AGGR98] is one the first approaches to subspace clustering. CLIQUE is a grid-based algorithm using an Apriori-like method to recursively navigate through the set of possible subspaces in a bottom-up way. The data space is first partitioned by an axis-parallel grid into equi-sized blocks of width ξ called units. Only units whose densities exceed a threshold τ are retained. Both ξ and τ are the input parameters of CLIQUE. The bottom-up approach of finding such dense units starts with 1- dimensional dense units. The recursive step from $(k-1)$-dimensional dense units to k-dimensional dense units takes $(k-1)$-dimensional dense units as candidates and generates the k-dimensional units by self-joining all candidates having the first $(k-2)$ dimensions in common. All generated candidates which are not dense are eliminated. For efficiency reasons, a pruning criterion called coverage is introduced to eliminate dense units lying in less "interesting" subspaces as soon as possible. For deciding whether a subspaces is interesting or not, the Minimum Description Length principle is used. Naturally this pruning bears the risk of missing some information. After generating all "interesting" dense units, clusters are found as a maximal set of connected dense units. For each k-dimensional subspace, CLIQUE takes all dense units of this subspace and computes disjoint sets of connected k-dimensional units. These sets are in a second step used to generate minimal cluster descriptions. This is done by covering each set of connected dense units with maximal regions and then determining the minimal cover.

ENCLUS (ENtropy-based CLUStering) [CFZ99] is a slight modification of CLIQUE. The major difference is the criterion used for subspace selection. The criterion of ENCLUS is based on entropy computation of a discrete random variable. The entropy of any subspace S is high when the points are uniformly distributed in S whereas it is lower the more closely the points in S are packed. Subspaces with an entropy below an input threshold ω are considered as good for clustering. A monotonicity criterion is presented to be used for a similar bottom-up algorithm as in CLIQUE.

MAFIA (Merging of Adaptive Finite IntervAls) [GNC99] is a more significant modification of CLIQUE. MAFIA uses adaptive, variable-sized grids in each dimension. A dedicated technique based on histograms which aims at merging grid cells is used to reduce the number of bins compared to CLIQUE. An input parameter α is used as a so-called cluster dominance factor to select bins which are α-times more densely populated (relative to their volume) than the average. The algorithm starts to produce such one-dimensional dense units as candidates and proceeds recursively to higher dimensions. In contrast to CLIQUE, MAFIA uses any two k-dimensional dense units to construct a new $(k+1)$-dimensional candidate as soon as they share an arbitrary $(k-1)$-face (not only first dimensions). As a consequence, the number of generated candidates is much larger compared to CLIQUE. Neighboring dense units are merged to form clusters. Redundant clusters, i.e. clusters that are true subsets of higher dimensional clusters, are removed.

SUBCLU (density-connected SUBspace CLUstering) [KKK04] overcomes the limitations of grid-based approaches like the dependence on the positioning of the grids. Instead of using grids the DBSCAN [EKSX96] cluster model of density-connected sets is used (cf. Chapter 3 for details on DBSCAN). SUBCLU is based on a bottom-up, greedy algorithm to detect the density-connected clusters in all subspaces of high-dimensional data. The algorithm starts with generating all 1-dimensional clusters w.r.t. the input parameters ε and μ by applying DBSCAN to each 1-dimensional subspace. Then, for each k-dimensional cluster it has to be checked iteratively if it is still existent in one ore more $(k+1)$-dimensional subspaces. For this purpose, all pairs of k-dimensional cluster having $(k-1)$ attributes in common are joined together to generate $(k+1)$-dimensional candidate subspaces. In the last step of the iteration the $(k+1)$-dimensional clusters are generated by applying DBSCAN to each cluster of one k-dimensional subspace of each $(k+1)$-dimensional candidate subspace. These steps are recursively executed as long as the set of k-dimensional subspaces containing clusters is not empty. Compared to the grid-based approaches SUBCLU achieves a better clustering quality but requires a higher runtime.

FIRES (FIlter REfinement Subspace clustering) [KKRW05] is a a general framework for efficient subspace clustering. It is generic in such a way that it works with all kinds of clustering notions. FIRES consists of the following three steps: preclustering, generation of subspace cluster approximations, and postprocessing. First, in the preclustering step, all 1-dimensional clusters called base-clusters are computed. This is similar to existing subspace clustering approaches and can be done using any clustering algorithm of choice. In a second step, the base-clusters are merged to find maximal-dimensional subspace cluster approximations. However, they are note merged in an Apriori style but by using an algorithm that scales at most quadratic w.r.t. the number of dimensions. As a last step, a postprocessing step can be applied to refine the cluster approximations retrieved after the second step.

5.2 Non-Overlapping Algorithms

PROCLUS (PROjected CLUStering) [APW+99] is a k-means like approach to subspace clustering assigning each point to one of k clusters. The number of clusters k and the average subspace dimension l are input parameters. PROCLUS proceeds in three phases: initialization, iteration, and cluster refinement. The initialization phase uses a greedy technique to select a set of potential medoids, such that each cluster is represented by at least one medoid. The iterative phase start with selecting k medoids as cluster representatives from the super set found in the initialization phase. Then iteratively the quality of clustering is improved by replacing bad medoids with randomly chosen new medoids. The quality of clustering is based on the average distances of the objects to their nearest medoids. The subspace of each cluster is found by examining the objects in the full-dimensional neighborhood of its medoid and computing some statistics. These statistics determine the relevant dimensions of the subspace of the cluster. The assignment of objects to the clusters is then based on the so-called Manhattan segmental distances relative to the assigned sets of dimensions. Finally, a cluster refinement is performed by computing new subspaces for each cluster and reassigning the objects to the clusters. PROCLUS suffers from the well-known problems of locally optimizing clustering methods and requires the user to specify the number k of clusters and the average dimensionality l of the subspaces of the clusters in advance. Since l is a rather sensitive parameter, PROCLUS usually has problems with subspace clusters of significantly different dimensionality. However, using a small number of representatives can cause PROCLUS to entirely miss some clusters. Thus, the cluster quality of PROCLUS is very sensitive to the chosen input parameters, which may be difficult to determine.

DOC (Density-based Optimal Projective Clustering) [PJAM02] proposes a mathematical definition of an "optimal projective cluster" along with a Monte Carlo algorithm to compute approximations of such optimal projective clusters. A projective cluster is defined as a pair (C, D) where C is a subset of the data set and D is a subset of the dimensions of the data space. Using the user specified input parameters ω and α, an optimal projective cluster (C, D) is given if C contains more than $\alpha\%$ of the data set and the projection of C into the subspace spanned by D must be contained in a hyper-cube of width ω whereas in all other dimensions $d \notin D$ the points in C are not contained in a hyper-cube of width ω. Another parameter β has to be specified that defines the balance between the number of points in C and the number of dimensions in D. The proposed algorithm DOC only finds approximations because it generates projected clusters of width 2ω. In addition, no assumption on the distribution of points inside such a hyper-cube is made. The reported projected clusters may contain additional noise objects (especially when the size of the projected cluster is considerably smaller than 2ω) and/or may miss some points that naturally belong to the projected cluster (especially when the size of the projected cluster is considerably larger than 2ω).

PreDeCon (subspace PREference weighted DEnsity CONnected clustering) [BKKK04] expands the density-based clustering notion of DBSCAN [EKSX96] to subspace clustering. PreDeCon builds for each point p a so-called subspace preference vector which reflects the variance of the points in the ε-neighborhood of p along each attribute. During the run of PreDeCon all points are either assigned to a certain cluster or marked as noise. For each point which is not yet classified, PreDeCon checks whether this point is a so-called preference weighted core point. Otherwise the point is marked as noise. To find a new cluster, PreDeCon starts with an arbitrary preference weighted core point p and adds all points that are preference weighted reachable from p to the current cluster. Then the algorithm continues with a point which has not yet been processed trying to expand a new cluster. PreDeCon has four input parameters, two density parameters ε and μ and two preference parameters λ and δ. The parameters ε and μ define a global density threshold by means of a radius ε and a minimum number of points μ in a region. The input parameter λ specifies the maximum subspace cluster dimensionality to be found, and parameter δ specifies the upper bound for the variance in an attribute. In fact, the dimensionality of the subspace clusters produced by PreDeCon are strongly biased towards λ. Thus, PreDeCon has problems with subspace clusters of significantly different dimensionality.

Chapter 6

HiSC: Finding Hierarchies of Subspace Clusters

Many traditional clustering algorithms are not applicable to high-dimensional feature spaces, because the clusters often exist only in specific subspaces of the original feature space. To cope with this problem, many subspace clustering algorithms have been proposed in recent years, which all aim at finding clusters in different subspaces within a data set. In this Chapter a new hierarchical subspace clustering algorithm, called HiSC (Hierarchical Subspace Clustering) is proposed that overcomes the following limitations of existing approaches:

1. HiSC can detect clusters in subspaces of significantly different dimensionality.

2. HiSC uncovers hierarchies of nested subspace clusters, i.e., the relationships of lower dimensional subspace clusters that are embedded within higher dimensional subspace clusters.

3. HiSC does not rely on a global clustering criterion.

4. The choice of parameters is considerably simplified compared to previous methods.

The rest of this Chapter is organized as follows. Section 6.1 contains the basic definitions for the main concepts of HiSC Section 6.2 provides the details of the new HiSC algorithm. The choice and impact of the input parameters are discussed in Section 6.3. Section 6.4 examines the runtime complexity of HiSC. Several comparative experiments, using synthetic and real-world data sets, show the performance and the effectivity of HiSC in Section 6.5. Parts of the material presented in this Chapter have been published in [ABK+06a].

6.1 Basic Definitions

Let $\mathcal{D}$ be a data set of n normalized feature vectors of dimensionality d ($\mathcal{D} \subseteq \mathbb{R}^d$). Let $\mathcal{A} = \{A_1, \ldots, A_d\}$ be the set of all attributes A_i of $\mathcal{D}$. Any subset $S \subseteq \mathcal{A}$ is called a *subspace*. The *projection* of an object $p \in \mathcal{D}$ into a subspace $S \subseteq \mathcal{A}$ is denoted by $\pi_S(p)$. The distance function between the feature vectors of $\mathcal{D}$ is denoted by $\textsc{Dist}$. It is assumed that $\textsc{Dist}$ is one of the L_p-norms. The k-nearest neighbors of an object $p \in DB$ for any $k \in \ NN^+$ are denoted by $\mathrm{NN}_k(p)$. More formally, the set of k-nearest neighbors of an object op is the smallest set $\mathrm{NN}_k(p) \subseteq \mathcal{D}$ that contains at least k objects from $\mathcal{D}$ such that

$$\forall o \in \mathrm{NN}_k(p), \forall o' \in \mathcal{D} - \mathrm{NN}_k(p) : \textsc{Dist}(p, o) < \textsc{Dist}(p, o').$$

In general, hierarchical clustering methods are able to find hierarchies of clusters which are nested into each other, i.e., weaker clusters in which some stronger clusters are contained. The hierarchical density-based clustering method OPTICS [ABKS99], for example, is able to detect clusters of higher density which are nested in clusters of lower but still high density. Adapting these ideas to subspace clustering, the user is interested in detecting subspace clusters of lower dimensionality which are contained in subspace clusters of higher dimensionality. The general idea is to evaluate whether two points are contained in a common subspace cluster. For example, two points that belong to two different 1-dimensional subspace cluster may also be contained together in a 2-dimensional cluster that consists of the two 1-dimensional projections. This verification is performed with a special distance measure called *subspace distance* which is introduced in Section 6.2. This distance results in a small value whenever two points are in a common low-dimensional subspace cluster, whereas the subspace distance is high if both points are in a common high-dimensional subspace cluster or are not in a subspace cluster at all. Therefore, the strategy is to merge those points into common clusters which have small subspace distances. A hierarchy of subspace clusters means that clusters containing objects having small subspace distances are nested in clusters containing objects having higher subspace distances to each other.

In order to define the already mentioned subspace distance, a *local subspace dimensionality* is assigned to each point of the data set in a preprocessing step. The local subspace dimensionality of a point represents the *subspace preference* of its local neighborhood, i.e., it reflects those attributes having a small variance in the local neighborhood of the point. Since a hierarchical approach is followed, the definition of the local neighborhood of a point does not rely on range queries as proposed in previous approaches such as in [BKKK04]. Rather, the k-nearest neighbors are used as local neighborhood of a point p, denoted by $\mathrm{NN}_k(p)$, to determine the variance in the local neighborhood of p.

Definition 6.1 (variance of the local neighborhood of a point).
The variance of the local neighborhood *of a point $p \in \mathcal{D}$ from p along an attribute $A_i \in \mathcal{A}$, denoted by $\textsc{Var}_{A_i}(NN_k(p))$, is defined as follows:*

$$\textsc{Var}_{A_i}(NN_k(p)) = \frac{\sum\limits_{q \in NN_k(p)} \left(\pi_{\{A_i\}}(q) - \pi_{\{A_i\}}(p)\right)^2}{|NN_k(p)|}.$$

Intuitively, the local subspace dimensionality is the number of attributes with high variance within the local neighborhood. Similar to [BKKK04], a local subspace preference vector is assigned to each point, indicating attributes with high and low variance within their local neighborhood.

Definition 6.2 (local subspace preference vector of a point).
Let $\alpha \in \mathbb{R}$ be a threshold value. The local subspace preference vector *of a*

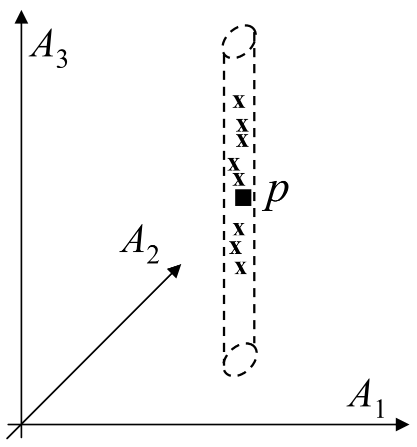


Figure 6.1: Visualization of the local subspace dimensionality of a point.

point $p \in \mathcal{D}$, denoted by $w_p = (w_p^1, \ldots, w_p^d)^{\mathbf{T}}$, is defined as

$$w_p^i = \begin{cases} 0 & \textit{if} \quad \textsc{Var}_{A_i}(NN_k(p)) > \alpha \\ 1 & \textit{if} \quad \textsc{Var}_{A_i}(NN_k(p)) \leq \alpha \end{cases} \textit{for } i = 1, \ldots, d.$$

The local subspace dimensionality of a point can now be defined as follows.

Definition 6.3 (local subspace dimensionality of a point).
The local subspace dimensionality λ_p *of a point $p \in \mathcal{D}$ is the number of zero-values in the local subspace preference vector of p, w_p, formally:*

$$\lambda_p = \sum_{i=1}^{d} \begin{cases} 1 & \textit{if} \quad w_p^i = 0 \\ 0 & \textit{if} \quad w_p^i = 1 \end{cases} .$$

An example is visualized in Figure 6.1. The 9-nearest neighbors of the 3-dimensional point p exhibit a 1-dimensional subspace cluster spanned by the attribute A_3, i.e., the variance of the local neighborhood of p has a high value along attribute A_3, whereas it has a low value along attributes A_1 and A_2. Consequently, $w_p = (1, 1, 0)^{\mathbf{T}}$ and $\lambda_p = 1$.

6.2 Algorithm HiSC

Once the points of the data set have been associated to a local subspace dimensionality and to a local subspace preference vector, the main concept of the hierarchical subspace clustering algorithm HiSC can be explained. Conventional hierarchical clustering algorithms like SLINK [Sib73] or OPTICS [ABKS99] work as follows: They keep two separate sets of points. The first set contains points which were already placed in the cluster structure and the second set consists of points which have not been processed already. In each step, one point of the latter set is selected and placed in the first set. The algorithm always selects that point which minimizes the distance to any of the points in the first set. By this selection strategy, the algorithm tries to extend the current cluster hierarchy as close to the bottom of the hierarchy as possible.

This paradigm will be adapted to the context of hierarchical subspace clustering where the hierarchy is a containment hierarchy of the subspaces. Two or more 1-dimensional subspace clusters may together form a 2-dimensional subspace cluster and so on. This behavior will be simulated by defining a similarity measure between two points which assigns a distance of 1, if these two points share a common 1-dimensional subspace cluster. If they share a common 2-dimensional subspace cluster, they have a distance of 2, etc. Sharing a common subspace cluster may imply different consequences: Both points may be associated to the same 2-dimensional subspace cluster, or both points may be associated to different 1-dimensional subspace clusters that intersect at some point or are parallel (but not skew).

If a distance measure with the properties mentioned before is assigned to a pair of points, the well-known hierarchical clustering algorithms can be used in general. Intuitively, the distance measure between two points corresponds to the dimensionality of the data space which is spanned by the attributes of high variance of the neighborhoods of the two points. Firstly, a definition of the *local subspace dimensionality of a pair of points* $\lambda(p, q)$ which follows the intuition of the spanned subspace is given. Secondly, the subspace distance measure based on these concepts will be defined. In fact, the subspace dimensionality is the most important component of the distance measure.

Definition 6.4 (local subspace preference vector of a pair of points). *The* local subspace preference vector $w(p,q) = (w^1(p,q), \ldots, w^d(p,q))$ *of a pair of points* $p, q \in \mathcal{D}$*, representing the attributes with low and high variance of the combined subspace is the attribute wise AND-conjunction of the local subspace preference vector* w_p *of* p *and the local subspace preference vector* w_q *of* q*, formally:*

$$w^i(p,q) = \begin{cases} 1 & \text{if } w_p^i = 1 \wedge w_q^i = 1 \\ 0 & \text{else} \end{cases} \quad \text{for } i = 1, \ldots, d.$$

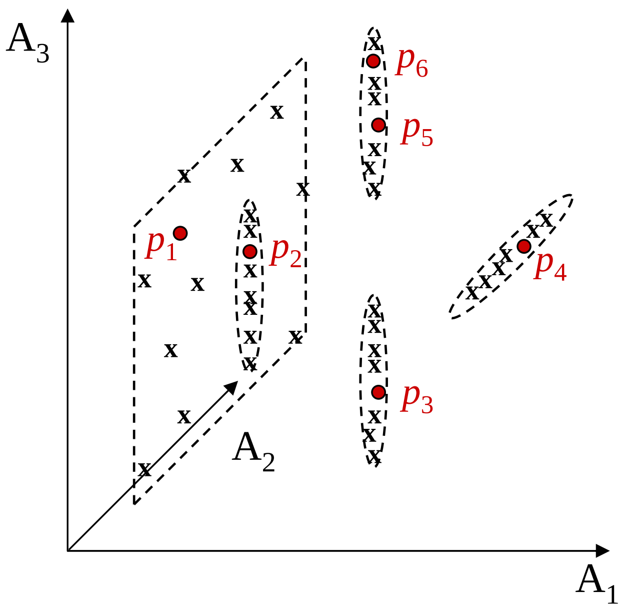
The local subspace dimensionality of a pair of points can now be defined as follows:

Definition 6.5 (local subspace dimensionality of a pair of points). *The* local subspace dimensionality *between two points* $p, q \in \mathcal{D}$*, denoted by* $\lambda(p,q)$*, is the number of zero-values in the local subspace preference vector of* p *and* q*,* $w(p,q)$*, formally:*

$$\lambda(p,q) = \sum_{i=1}^{d} \begin{cases} 1 & \text{if} \quad w^i(p,q) = 0 \\ 0 & \text{if} \quad w^i(p,q) = 1 \end{cases} .$$

An example is visualized in Figure 6.2. The upper figure shows on the left hand side a 3-dimensional data space with three 1-dimensional subspace clusters, one of them embedded in a fourth 2-dimensional subspace cluster. The subspace dimension of point p_1 which is a member of the 2-dimensional subspace cluster is $\lambda_{p_1} = 2$, whereas the subspace dimensionality of the other five highlighted points $p_2, \ldots, p_6$ is 1. The combined subspace dimensionalities are depicted in the lower figure on the left hand side.

A first approach defines the subspace distance between two points p and q as the local subspace dimensionality $\lambda(p,q)$. A slight extension for points that have the same local subspace preference vector, but do not belong to the same subspace cluster is needed. For example, in Figure 6.2, the points



	$\lambda(p_i, p_j)$						$\text{SDIST}(p_i, p_j).d_1$					
	p_1	p_2	p_3	p_4	p_5	p_6	p_1	p_2	p_3	p_4	p_5	p_6
p_1		2	2	2	2	2		2	3	3	3	3
p_2	2		1	2	1	1	2		2	3	2	2
p_3	2	1		2	1	1	3	2		3	1	1
p_4	2	2	2		2	2	3	3	3		3	3
p_5	2	1	1	2		1	3	2	1	3		1
p_6	2	1	1	2	1		3	2	1	3	1	

Figure 6.2: Visualization of the subspace dimensionality and subspace distance of pairs of points.

p_2 and p_3 will have the same preference vector and therefore, have a common subspace dimensionality of 1, indicating that they are in a common 1-dimensional subspace cluster. Obviously, this is not the case. Though the two 1-dimensional subspace clusters the points belong to are in parallel projections, the two clusters span together a 2-dimensional subspace because they are considerably far apart along at least one attribute of low variance. On the other hand, the subspace clusters of points p_3 and p_5 span together a 1-dimensional subspace because they are (though considerably far apart along the attribute of high variance) compactly located along the attributes with low variance.

In order to formalize these intuitions, the distance between the points along the attributes of low variance has to be checked. If this distance, which can be evaluated by a simple weighted Euclidean distance using the common preference vector of p and q as weighting vector, exceeds α, the points or corresponding clusters do not belong to the same cluster but belong to different parallel clusters. The threshold value α, which indicates attributes with low and high variance within their local neighborhood in Definition 6.2, controls also the degree of jitter of the subspace clusters.

As $\lambda(p, q) \in \mathbb{N}$, many distances between different point pairs are identical. Therefore, there are many tie situations during clustering. These tie situations are resolved by additionally considering the Euclidean distance within a subspace cluster as a secondary criterion. This means, inside a subspace cluster (if there are no nested lower dimensional subspace clusters), the points are clustered in the same way as using a conventional hierarchical clustering method. The Euclidean distance between p and q hereby is weighted by the inverse of the combined preference vector $w(p, q)$, as given in Definition 6.4. The inverse of the combined preference vector $w(p, q)$, denoted by $\bar{w}(p, q) = (\bar{w}^1(p, q), \ldots, \bar{w}^d(p, q))$, is defined as

$$\bar{w}^i(p, q) = \begin{cases} 0 & \text{if} \quad w^i(p, q) = 1 \\ 1 & \text{if} \quad w^i(p, q) = 0 \end{cases} \quad \text{for } i = 1, \ldots, d.$$

This inverse subspace preference vector $\bar{w}(p, q)$ weights the distance along attributes spanning the cluster with 1, the distance along any other attribute is weighted with 0. Formally:

Definition 6.6 (subspace distance).
Let $v = (v^1, \ldots, v^d)^{\mathbf{T}}$ be a d-dimensional vector, and

$$\textsc{Dist}_v(p, q) = \sqrt{\sum_{i=1}^{d} v^i (\pi_{\{A_i\}}(p) - \pi_{\{A_i\}}(q))^2}$$

be the weighted Euclidean distance w.r.t. v between two points $p, q \in \mathcal{D}$. The subspace distance *between p and q, denoted by* $\textsc{SDist}(p,q) = (d_1, d_2)$*, is a pair consisting of the following two values:*

$$\begin{aligned} d_1 &= \lambda(p,q) + \begin{cases} 1 & \textit{if } \textsc{Dist}_{w(p,q)}(p,q) > \alpha \\ 0 & \textit{else,} \end{cases} \\ d_2 &= \textsc{Dist}_{\bar{w}(p,q)}(p,q). \end{aligned}$$

$\textsc{SDist}(p,q) \leq \textsc{SDist}(r,s)$ *if one of the following conditions hold:*

1. $\textsc{SDist}(p,q).d_1 < \textsc{SDist}(r,s).d_1$

2. $\textsc{SDist}(p,q).d_1 = \textsc{SDist}(r,s).d_1$ *and* $\textsc{SDist}(p,q).d_2 \leq \textsc{SDist}(r,s).d_2$

As discussed above, d_1 corresponds to the local subspace dimensionality of p and q, taking special care in case of parallel clusters. The value d_2 corresponds to the weighted Euclidean distance between p and q, where the inverse of the combined preference vector is used, $\bar{w}(p,q)$, as weighting vector. An example can be seen in Figure 6.2, where the values of d_1 of the corresponding subspace distances between the points are depicted in the lower figure on the right hand side.

Using the subspace distance defined above as a distance measure, basically every hierarchical or even non-hierarchical clustering algorithm which is based on distance comparisons can be applied. Examples for such algorithms are Single-Link [Sib73] and its variant Complete-Link [Def77], and the density-based clustering methods DBSCAN [EKSX96] (non-hierarchical) and OPTICS [ABKS99].

HiSC follows a Single-Link based approach and selects in each step of the algorithm that point p having the minimum subspace distance to any already processed point. For that purpose, each object p has an additional attribute $p.\textsc{MinSDist}$ that holds the minimum subspace distance to any object processed before p. The main data structure is a priority queue organized as a heap which stores all points according to their minimum subspace distances in ascending order. Initially, all points of the data set are added

```
algorithm HiSC(Database $\mathcal{D}$, Integer $k$, Real $\alpha$)
    initialize empty subspace cluster order $co$;
    initialize empty priority queue $pq$ ordered by $\textsc{MinSDist}$;
    for each $p \in \mathcal{D}$ do
        compute $w_p$ w.r.t. parameters $k$ and $\alpha$;
        $p.\textsc{MinSDist} = \infty$;
        $pq$.insert($p$);
    end for
    while $pq \neq \emptyset$ do
        $p := pq$.next();
        $co$.add($p$);
        for each $q \in pq$ do
            if $\textsc{SDist}(p, q) < q.\textsc{MinSDist}$ then
                $q.\textsc{MinSDist} := \textsc{SDist}(p, q)$;
                $pq$.decrease($q$);
            end if
        end for
    end while
    return $co$;
end.
```

Figure 6.3: The HiSC algorithm.

to the priority queue with a minimum subspace distance of ∞ and the local subspace preference vector of each point is computed. The next point p to be processed is always the first object in the priority queue. Then, the minimum subspace distances of objects q which are still remaining in the priority queue are updated if their according values decrease. In this way, a special order of the data set according to its subspace-based clustering structure is generated, the so-called *subspace cluster order*. Using a visualization technique similar to that of OPTICS [ABKS99], the subspace cluster order can be displayed in a subspace distance diagram. Such a subspace distance diagram consists of the subspace distance values on the y-axis of all points, plotted in the order which HiSC produces on the x-axis. The result is a visualization of the subspace clustering structure of the data set which is very easy to comprehend and interpret. The “valleys” in the plot represent the subspace clusters, since points within a subspace cluster typically have lower subspace distances than points outside of a subspace cluster. The complete integration of the subspace distance measure into the algorithm HiSC can be seen in Figure 6.3.

6.3 Input Parameters

HiSC has two input parameters. First, the parameter k specifies the locality of the neighborhood from which the local subspace dimensionality of each point is determined. Obviously, this parameter is rather critical because if it is chosen too large, the local subspace preference may be blurred by noise points, whereas if it is chosen too small, there may not be a clear subspace preference observable, although existing. However, in the experiments, choosing k in the range between 10 and 20 turned out to produce very stable and robust results.

Second, the parameter α is important for specifying the attributes with low and high variance within their local neighborhood and therefore, α affects the computation of the local subspace dimensionality of each point. In fact, attributes where the variance of the k-nearest neighbors of a point is below α are relevant for the subspace preference of the point. In the experiments, it turned out that HiSC is quite robust against the choice of α as long as α is chosen between 0.1% and 0.5% of the attribute range, i.e., the maximum attribute value. However, if subspace clusters having a lot of jitter are expected, α can be increased accordingly.

6.4 Runtime Complexity

Let n be the number of data points and d be the dimensionality of the data space. In the first loop the local subspace dimensionalities and local preference vectors are precomputed which requires the determination of the k-nearest neighbors of each object. Assuming a suitable spatial index structure, this can be done in $O(\log n \cdot d)$ time. Since this step is done for each object in the data set, the runtime complexity for the preprocessing step results in $O(n \cdot \log n \cdot d)$ if a spatial access method to support k-nearest neighbor queries exists. If no such spatial index is at hand, the complexity of the preprocessing step is $O(n^2 \cdot d)$.

During the run of HiSC, for each point p of the data set its subspace distance to all remaining points q in the priority queue has to be evaluated. This requires first the determination of the subspace dimensionality of p and q in form of a simple logical AND-conjunction on the subspace preference vectors of p and q which has a complexity of $O(d)$. Secondly, the weighted distance between p and q has to be computed, which also can be done in $O(d)$. Thus, the complexity of the main loop of HiSC yields in $O(n^2 \cdot d)$ time.

Overall, the complete runtime complexity of HiSC results in $O(n^2 \cdot d)$.

6.5 Experimental Evaluation

All experiments have been performed on a workstation with a $2 \cdot 64$-bit 2.6 GHz CPU and 16 GB main memory. All evaluated methods have been implemented in Java. In all experiments, the input parameters of all methods have been optimized in terms of quality and the best results have been reported in order to achieve a fair comparison.

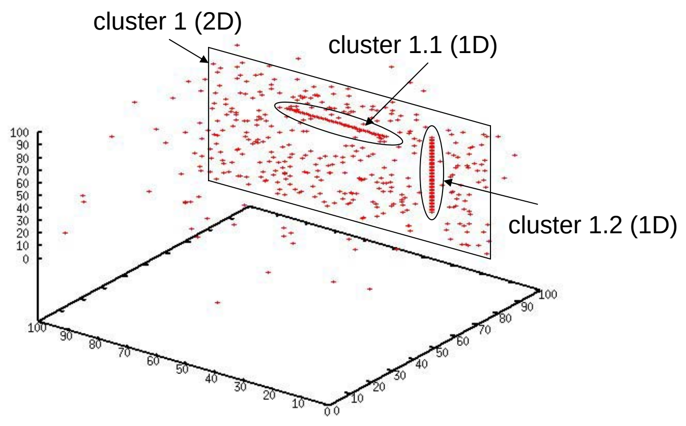


Figure 6.4: Data set "DS1".

6.5.1 Effectivity

Synthetic Data Sets. First, HiSC has been evaluated on several synthetic data sets. Exemplary, the results on two data sets named "DS1" and "DS2" are shown. The synthetic data sets contain 3- and 20-dimensional objects grouped in hierarchical subspace clusters and additional noise points. The attribute values of all synthetic data sets are in the range of 0.0 to 100.0.

Data set "DS1" (cf. Figure 6.4) contains 3-dimensional points grouped in three hierarchical subspace clusters and noise. Two 1-dimensional subspace clusters (cluster 1.1 and cluster 1.2) are both embedded within a 2-dimensional subspace cluster (cluster 1). The subspace distance diagram produced by HiSC applied to "DS1" with parameter setting $\alpha = 0.0001$ and $k = 11$ is depicted in Figure 6.5. As it can be seen, the complete hierarchical clustering structure can be obtained from the resulting reachability plot. In particular, the nested clustering structure of the two 1-dimensional subspace clusters embedded within the 2-dimensional subspace cluster can be seen at first glance.

For comparison, PreDeCon and PROCLUS have also been applied to data set "DS1". As expected, PreDeCon can either detect the 1-dimensional subspace cluster or the 2-dimensional subspace clusters, but not both in one single run. The results of PreDeCon with different settings for parameter λ,

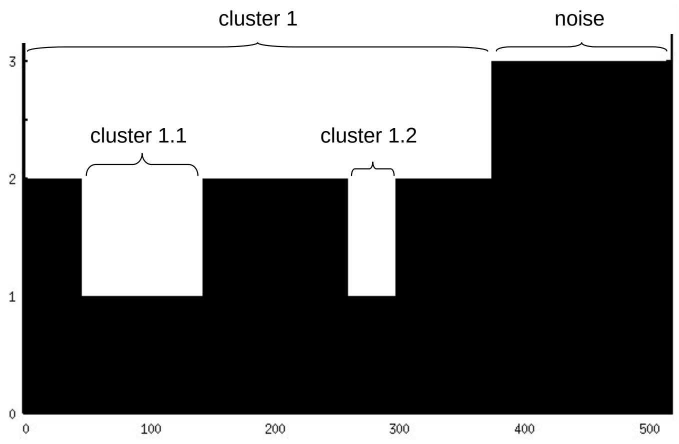
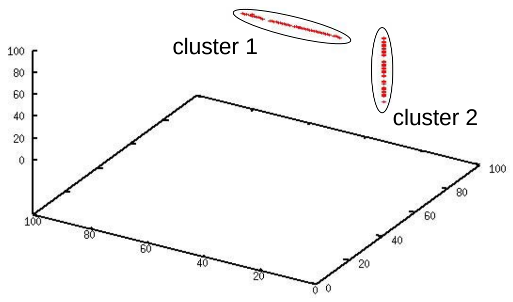


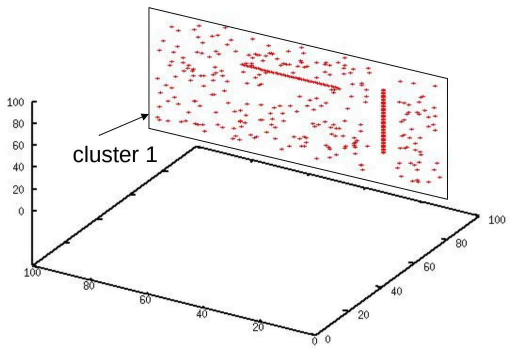
Figure 6.5: Result of HiSC on "DS1".

which determines the subspace dimensionality of the clusters to be found, are depicted in Figure 6.6. The left Figure 6.6(a) shows the two 1-dimensional subspace clusters found by PreDeCon with parameter setting $\varepsilon = 0.1, \mu = 20, \lambda = 1, \delta = 0.001$. In this run, the 2-dimensional plane was classified as noise. In the right Figure 6.6(b) the 2-dimensional subspace cluster detected by PreDeCon with parameter setting $\varepsilon = 0.25, \mu = 20, \lambda = 2, \delta = 0.001$ is depicted. This cluster includes the two 1-dimensional subspace clusters, thus in both runs PreDeCon was not able to detect the hierarchies and all subspace clusters in "DS1". As it can be seen in Figure 6.7, PROCLUS ($k = 3, l = 3$) failed completely in finding the subspace clusters in "DS1".

Data set "DS2" is a 20-dimensional data set containing three subspace clusters of significantly different dimensionality and noise: subspace cluster 1 is a 15-dimensional subspace cluster, subspace cluster 2 is 10-dimensional, and subspace cluster 3 is a 5-dimensional subspace cluster. The resulting subspace distance diagram produced by HiSC on "DS2" ($\alpha = 0.00001, k = 60$) is shown in Figure 6.8. HiSC has no problems detecting the three subspace clusters of considerably different dimensionality. The clusters can again be visually explored at first glance. Again, PreDeCon and PROCLUS have also been applied to data sets "DS2". Like in the first experiment, both algorithms failed to detect the hierarchies and all subspace clusters of significantly different dimensionality.



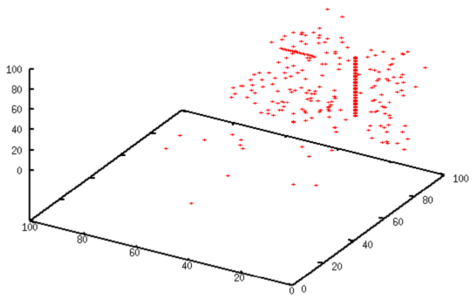
(a) Result of PreDeCon with $\lambda = 1$.

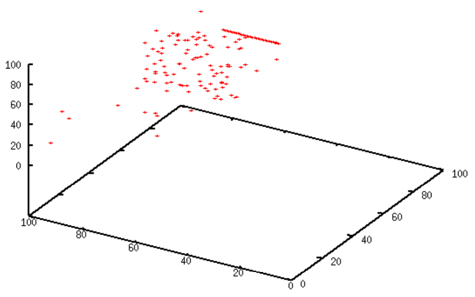


(b) Result of PreDeCon with $\lambda = 2$.

Figure 6.6: Results of PreDeCon with different λ-parameter settings on "DS1".

Real-world Data Set. HiSC has been applied to a real-world data set named "Metabolome", containing metabolic screening data of 2,000 newborns. For each newborn, the blood-concentration of 43 different metabolites were measured. Thus, the data set is 43-dimensional containing 2,000 objects. The newborns are labeled by one of three categories. The healthy patients are marked by "control", newborns suffering phenylketonuria (a well-known metabolic disease) are labeled with "PKU", and newborns suffering any other metabolic disease are labeled with "others". The resulting subspace distance diagram HiSC generates when applied to this data set is visualized in Figure 6.9. As it can be seen, HiSC produced a large hierarchy of 17-dimensional to 25-dimensional subspace clusters nested into each other. All these subspace clusters contain approximately 98% newborns marked with "control". A second hierarchy of nested subspace clusters contains only newborns marked with "PKU". The rest is a mix of all three categories. However, it can be observed that the newborns marked with "other" are mainly at the end of the subspace cluster ordering having rather high subspace dimensionalities. In summary, it can be stated that HiSC is able to clearly separate the vast majority of the three classes present in the data set.


(a) PROCLUS - cluster 1.


(b) PROCLUS - cluster 2.

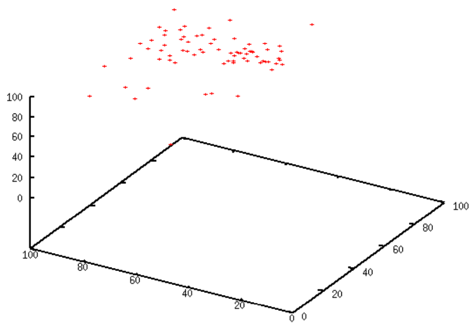

(c) PROCLUS - cluster 3.

Figure 6.7: Results of PROCLUS on “DS1”.

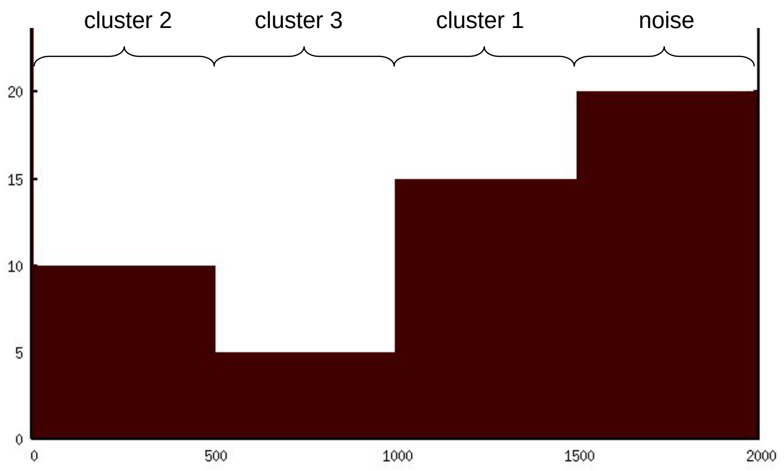


Figure 6.8: Results of HiSC on “DS2”.

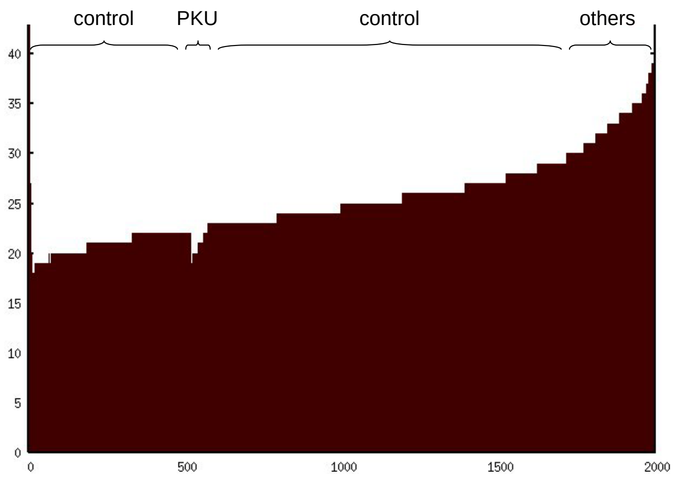


Figure 6.9: Results of HiSC on "Metabolome" data.

6.5.2 Scalability

The scalability of HiSC w.r.t. the dimensionality of the data set is depicted in Figure 6.10. The experiments were obtained by using 10 synthetic data sets of 10,000 objects with varying dimensionality of $d = 10, 20, 30, \ldots, d_{max} = 100$. For each data set, the objects were equally distributed over 10 subspace clusters, where the single attributes have values in the range of $[0.0, 1.0]$. The result shows a linear increase of runtime when increasing the dimensionality of the data set. The parameters for HiSC were set to $k = 3 \cdot d_{max} = 300$ and $\alpha = 0.1\%$ of the attribute range, i.e., $\alpha = 0.001$. This experiment confirms the theoretically determined runtime complexity of $O(n^2 \cdot d)$ (cf. Section 6.4).

A similar observation can be made when evaluating the scalability of HiSC w.r.t. the data set size (cf. Figure 6.11). The experiment was run on a set of 6 10-dimensional synthetic data sets with varying number of objects ranging from 50,000 to 300,000. The objects are equally distributed over nine subspace clusters of subspace dimensionality $\lambda = 1, \ldots, 9$ and noise, where the attribute values are in the range of 0.0 to 1.0. The parameters for HiSC were set to $k = 3 \cdot d = 15$ and $\alpha = 0.1\%$ of the attribute range, i.e., $\alpha = 0.001$. As it can be seen, HiSC scales quadratic w.r.t. the number of tuples in the data set. Again, this experiments confirms the theoretically

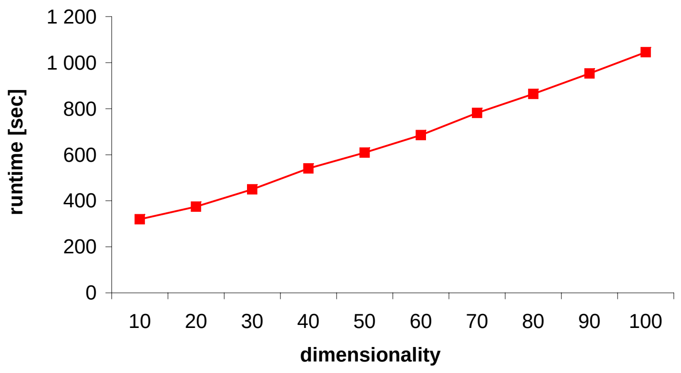


Figure 6.10: Scalability of HiSC w.r.t. the dimensionality.

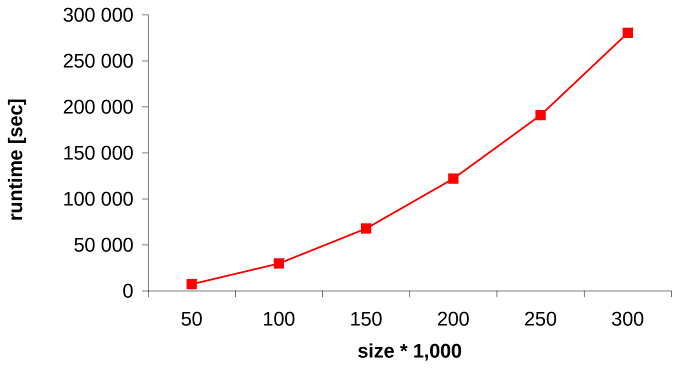


Figure 6.11: Scalability of HiSC w.r.t. the size.

determined runtime complexity of $O(n^2 \cdot d)$.

In summary, the scalability experiments show that HiSC scales well also for large and high-dimensional data sets.

Chapter 7

DiSH: Detecting and Visualizing Complex Hierarchies of Subspace Clusters

Existing subspace clustering algorithms such as [APW$^+$99, PJAM02, BKKK04] assign each point to a unique subspace cluster or noise. Usually, those methods do not produce any information on the hierarchical relationships among the detected subspaces. The only approach to find some special cases of subspace cluster hierarchies introduced so far is HiSC (cf. Chapter 6). However, HiSC is limited by the following severe drawbacks. First, HiSC usually assumes that if a point p belongs to a projected cluster C, then C must be visible in the local neighborhood of p in the *entire* feature space. Obviously, this is a quite unrealistic assumption. If p belongs to a projected cluster and the local neighborhood of p in the entire feature space does not exhibit this projection, HiSC will not assign p to its correct cluster. Second, the hierarchy detected by HiSC is limited to single inclusions which can be visualized by a tree representation such as a dendrogram. As discussed above, hierarchies of subspace clusters may also exhibit multiple inclusions. To visualize such more complex relationships among subspace clusters, graph representations are needed rather than tree representations. Third, HiSC uses a Single-Link approach for clustering and, thus, suffers from the so-called Single-Link effect, which means that a single noise object bridging the gap between two actual subspace clusters can hamper the algorithm in detecting the correct subspace clustering structure.

To overcome these limitations, the hierarchical subspace clustering algorithm DiSH (Detecting Subspace Cluster Hierarchies) is proposed in this Chapter. DiSH overcomes the limitations of existing subspace clustering approaches mentioned before and claims the following contributions:

1. DiSH assigns points to their correct subspace clusters even if this projection is not visible in the local neighborhoods of the points in the entire feature space.

2. DiSH applies a density-based hierarchical approach similar to OPTICS [ABKS99] to the subspace clustering problem. Thus, DiSH avoids Single-Link effects. Furthermore, it is able to determine hierarchies of nested subspace clusters containing single and multiple inclusions.

3. DiSH can detect subspace clusters of significantly different subspace dimensionality.

4. DiSH computes a clear and intuitive graph representation of the result such that the complete relationships among subspace clusters are presented and the entire hierarchical subspace clustering structure can be explored at a glance.

Section 7.1 introduces the basic definitions and explains the necessary preprocessing steps to be performed before applying DiSH. Section 7.2 introduces the main concepts of the new DiSH algorithm in detail. The choice and impact of the input parameters are discussed in Section 7.3, and Section 7.4 examines the runtime complexity of DiSH. The generation of a clear representation of the complex subspace cluster hierarchy in a so-called subspace clustering graph is shown in Section 7.5. Several comparative experiments presented in Section 7.6 using synthetic and real-world data sets show the efficiency and the effectivity of DiSH. The basic ideas contained in this Chapter have been published in [ABK+07a].

7.1 Basic Definitions

Let $\mathcal{D} \subseteq \mathbb{R}^d$ be a data set of n normalized feature vectors and $\mathcal{A} = \{A_1, \ldots, A_d\}$ be the set of attributes of $\mathcal{D}$. For any subspace $S \subseteq \mathcal{A}$, $\pi_S(p)$ denotes the projection of object $p \in \mathcal{D}$ into S. Furthermore, it is assumed that $\textsc{Dist}$ is a distance function applicable to any subspace $S \subseteq \mathcal{A}$, denoted by $\textsc{Dist}^S$. For instance, when using the Euclidean distance for $p, q \in \mathcal{D}$,

$$\textsc{Dist}^S(p,q) = \sqrt{\sum_{A_i \in S} \left(\pi_{\{A_i\}}(p) - \pi_{\{A_i\}}(q)\right)^2}.$$

The key idea of DiSH is similar to HiSC and is based on the definition of the so-called *subspace distance* that assigns small values if two points are in a common low-dimensional subspace cluster, and high values if two points are in a common high-dimensional subspace cluster or are not in a subspace cluster at all. Again, subspace clusters with small subspace distances are embedded within clusters with higher subspace distances.

In contrast to HiSC, the subspace distance used in the DiSH algorithm is not based on the local neighborhoods of the points, but on the global neighborhoods in each dimension. First, for each point $p \in \mathcal{D}$ the subspace dimensionality representing the dimensionality of that subspace cluster in which p fits best is computed. Thereby, it is assumed that the "best" projection for clustering p is the subspace with the highest dimensionality providing the most information. In case of tie-situations, p will be assigned to the larger subspace cluster, containing more points in the neighborhood of p w.r.t. the subspace. The subspace dimensionality of a point p is determined by searching for dimensions of low variance in the neighborhood of p. Intuitively, a low variance along one attribute indicates that if the points are projected on that dimension, p lies within a dense area. To detect dimensions of low variance, i.e., high density, the ε-neighborhood of a point p in each dimension can be used.

Definition 7.1 (ε-neighborhood of a point w.r.t. a subspace).
Let $\varepsilon \in \mathbb{R}^+$ be a threshold value. The ε-neighborhood of a point $p \in \mathcal{D}$ w.r.t. subspace $S \subseteq \mathcal{A}$, *denoted by $N_\varepsilon^S(p)$, is defined as follows:*

$$N_\varepsilon^S(p) = \{q \,|\, q \in \mathcal{D} \wedge \textsc{Dist}^S(p,q) \leq \varepsilon\}.$$

An attribute-wise ε-range query for each $A_i \in \mathcal{A}$ yields a simple way to assign a predicate to an attribute for a certain object p. If only few points are found within the ε-neighborhood in attribute A_i most of the attributes will be distributed over a broader range and, thus the variance around p in attribute A_i will be relatively high. Therefore, this attribute does not participate in a subspace that is relevant to any cluster to which p could possibly belong. Otherwise, if $\mathrm{N}_\varepsilon^{\{A_i\}}(p)$ contains at least μ objects, the attribute A_i will be a candidate for a subspace containing a cluster including object p.

Having determined the candidate attributes that might span the subspace S_p in which object p is clustered, these attributes have to be combined in a suitable way. In fact, the problem of finding the correct subspace is equivalent to finding the "correct" item-subset of a set of items and thus, is equivalent to itemset mining. The items correspond to (candidate) attributes. Each point $q \in \mathcal{D} - \{p\}$ represents a transaction $T^p(q)$ with items A_i such that $q \in \mathrm{N}_\varepsilon^{\{A_i\}}(p)$, i.e., $T^p(q) = \{A_i \,|\, p \in \mathrm{N}_\varepsilon^{\{A_i\}}(p)\}$. The support of item A_i is defined by $sup(A_i) = |\mathrm{N}_\varepsilon^{\{A_i\}}(p)|$. Obviously, the support is anti-monotonic, i.e., $sup(T) \leq sup(S)$ for all $S \subseteq T \subseteq \mathcal{A}$. As a consequence, any frequent itemset mining algorithm can be used in order to determine the best subspace S_p of an object p, e.g., the Apriori-algorithm [AS94]. In particular, for an object $p \in \mathcal{D}$ the maximum frequent itemset representing the best subspace S_p is determined such that $|\mathrm{N}_\varepsilon^{S_p}(p)| \geq \mu$, i.e., $sup(S_p) \geq \mu$. As discussed above, if there are several such itemsets/subspaces, the one with the highest support is chosen. Then, the subspace preference vector w_p of object $p \in \mathcal{D}$ is defined as follows.

Definition 7.2 (subspace preference vector of a point).
Let S_p be the best subspace determined for object $p \in \mathcal{D}$, i.e., the highest

dimensional subspace S with $|N_\varepsilon^S(o)| \geq \mu$. In case of tie situations let S_p be the subspace containing the most objects in the ε-neighborhood of p w.r.t. S_p. The subspace preference vector *of p, denoted by $w_p = (w_p^1, \ldots, w_p^d)^{\mathbf{T}}$, is defined as*

$$w_p^i = \begin{cases} 1 & \textit{if} \quad A_i \in S_p \\ 0 & \textit{if} \quad A_i \notin S_p \end{cases} \quad \textit{for } i = 1, \ldots, d.$$

Obviously, the exhaustive search for the best subspace using the Apriori-algorithm or one of its variants is rather inefficient for high-dimensional data sets, especially when the dimensionality of the subspace clusters is also high. The worst case complexity of the Apriori-algorithm results in an exponential runtime of $O(2^d)$. To overcome this limitation, in the following a heuristics is proposed for determining the best subspace S_p for an object p which scales quadratic in the number of dimensions (cf. Section 7.4 for a detailed discussion).

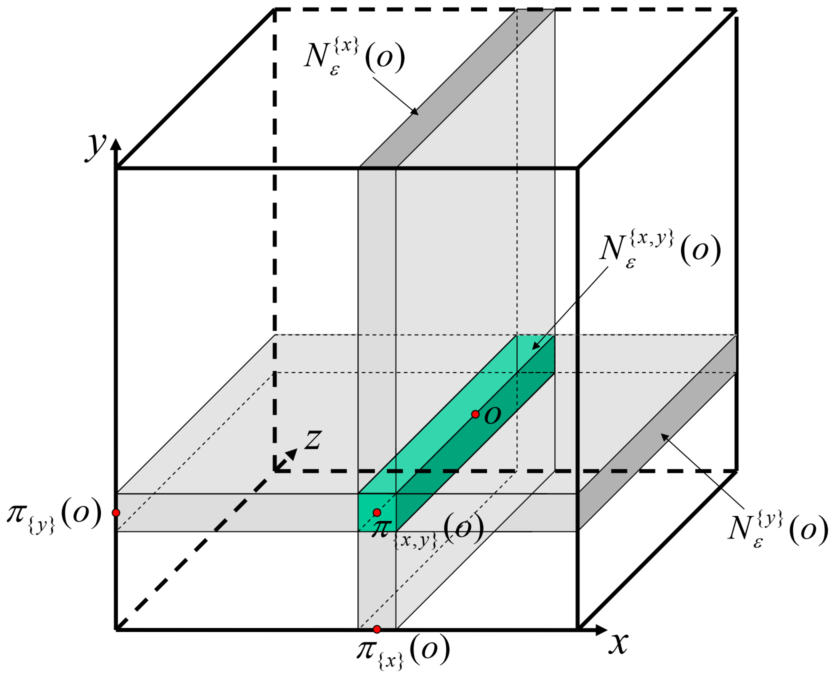
The method to determine the best subspace for an object p is depicted in Figure 7.1. First, the candidate attributes C_p of object p are determined by computing the ε-neighborhoods $\mathrm{N}_\varepsilon^{\{A_i\}}(p)$ of p in each attribute $A_i \in \mathcal{A}$. If $\mathrm{N}_\varepsilon^{\{A_i\}}(p)$ contains at least μ objects, attribute A_i will be added to the candidate set C_p. After determining the candidate attributes that might span S_p, simply a best-first search is used, starting with the attribute $A_i \in C_p$ where the number of objects in $\mathrm{N}_\varepsilon^{\{A_i\}}(p)$ is highest. This attribute is added to the subspace S_p and removed from the candidate set C_p. The intersection of the ε-neighborhoods of p in the attributes already assigned to S_p , denoted by I, is initialized with $\mathrm{N}_\varepsilon^{\{A_i\}}(p)$. Then, in the merging loop iteratively that attribute is merged to S_p that "fits best" to the attributes already belonging to S_p. The "best" attribute A_i is always that attribute of the candidate set C_p, where the intersection of the ε-neighborhood in A_i with I contains the most objects. If the intersection of I and $\mathrm{N}_\varepsilon^{\{A_i\}}(p)$ contains at least μ objects, attribute A_i is added to the subspace S_p and removed from the candidate set C_p. The intersection of the ε-neighborhoods of p in the already merged attributes is updated. The search for the best subspace S_p terminates if the candidate set is empty or $\mathrm{N}_\varepsilon^{A_i}(p)$ contains less than μ objects, i.e., there exists no more attribute $A_i \in C_p$ which can extend S_p to a subspace containing at

```
function determineBestSubspace(Object $p$, Real $\varepsilon$, Integer $\mu$)
    initialize empty subspace $S_p$;
    $C_p := \{A_i \mid A_i \in \mathcal{A} \wedge |\mathsf{N}_\varepsilon^{\{A_i\}}(p)| \geq \mu\}$;
    $A_i = \arg\max_{A_j \in C_p} \{|\mathsf{N}_\varepsilon^{\{A_j\}}(p)|\}$;
    $S_p$.add($A_i$);
    $C_p$.remove($A_i$);
    $I := \mathsf{N}_\varepsilon^{\{A_i\}}(p)$;
    while $C_p \neq \emptyset$ do
        $A_i = \arg\max_{A_j \in C_p} \{|I \cap \mathsf{N}_\varepsilon^{\{A_j\}}(p)|\}$;
        if $|I \cap \mathsf{N}_\varepsilon^{\{A_i\}}(p)| \geq \mu$ then
            $S_p$.add($A_i$);
            $C_p$.remove($A_i$);
            $I := I \cap \mathsf{N}_\varepsilon^{\{A_i\}}(p)$
        else
            return $S_p$;
        end if
    end while
    return $S_p$;
end.
```

Figure 7.1: The heuristics to determine the best subspace for an object.

least μ objects in the ε-neighborhood.

Using this heuristics to compute the best subspace S_p for $p \in \mathcal{D}$, the subspace preference vector of object p, w_p, can be determined as defined in Definition 7.2. Again, the assigned predicate for each attribute describes the preference for the corresponding subspace. Overall, a d-dimensional subspace preference vector is assigned to each point, containing “1” and “0” values that represent the above described predicates for each attribute. The attributes having a “1” predicate span the subspace of the cluster the point belongs to.



$N_\varepsilon^{\{x\}}(o)$
y
$N_\varepsilon^{\{x,y\}}(o)$
o
z
$\pi_{\{y\}}(o)$
$\pi_{\{x,y\}}(o)$
$N_\varepsilon^{\{y\}}(o)$
$\pi_{\{x\}}(o)$
x

Figure 7.2: Subspace selection for a point o.

The remaining attributes with “0” predicates are irrelevant for that subspace.

The subspace dimensionality of a point can now be defined as follows.

Definition 7.3 (subspace dimensionality of a point).
The subspace dimensionality λ_p *of a point* $p \in \mathcal{D}$ *is the number of zero-values in the subspace preference vector of* p, w_p, *formally:*

$$\lambda_p = \sum_{i=1}^{d} \begin{cases} 1 & \textit{if} \quad w_p^i = 0 \\ 0 & \textit{if} \quad w_p^i = 1 \end{cases} .$$

In the example in Figure 7.2 the ε-neighborhoods of the 3-dimensional point o in attributes x and y are shown by gray-shaded areas. Assuming that both of these areas contain at least μ points whereas the ε-neighborhood of o along z (not shown) contains less than μ points, o may participate in a subspace cluster that is projected into the subspace $\{x, y\}$. The cardinality of $\mathrm{N}_\varepsilon^{\{x,y\}}(o)$ has still to be tested, i.e., the cardinality of $\mathrm{N}_\varepsilon^{\{x\}}(p) \cap \mathrm{N}_\varepsilon^{\{y\}}(p)$. If this cardinality is also greater than or equal to μ, then $w_o = (1, 1, 0)^{\mathbf{T}}$ and

$\lambda_o = 1$. On the other hand, if $|\mathrm{N}_\varepsilon^{\{x\}}(o) \cap \mathrm{N}_\varepsilon^{\{y\}}(o)| < \mu$, the projection of the neighborhood of o into subspace $\{x, y\}$ is not dense. In other words, o may be part of several 1-dimensional subspace clusters, but none of these subspace clusters can be merged to form a higher dimensional subspace cluster. In that case, o will be classified as noise.

7.2 Algorithm DiSH

After assigning during a preprocessing step the subspace dimensionality and the subspace preference vector to each point p of the data set, the similarity measure between two points, called the *subspace distance*, can be defined analogously as in Chapter 6.2. First, the definitions of the subspace preference vector and the subspace dimensionality of a pair of points are given, then the distance measure is defined.

Definition 7.4 (subspace preference vector of a pair of points).
The subspace preference vector $w(p, q) = (w^1(p, q), \ldots, w^d(p, q))$ *of a pair of points $p, q \in \mathcal{D}$ representing the combined subspace of p and q is computed by the attribute wise logical AND-conjunction of the subspace preference vector w_p of p and the subspace preference vector w_q of q, formally:*

$$w^i(p, q) = \begin{cases} 1 & \text{if } w_p^i = 1 \wedge w_q^i = 1 \\ 0 & \text{else} \end{cases} \quad \text{for } i = 1, \ldots, d.$$

Definition 7.5 (subspace dimensionality of a pair of points).
The subspace dimensionality *between two points $p, q \in \mathcal{D}$, denoted by $\lambda(p, q)$, is the number of zero-values in the subspace preference vector of p and q, $w(p, q)$, formally:*

$$\lambda(p, q) = \sum_{i=1}^{d} \begin{cases} 1 & \text{if} \quad w^i(p, q) = 0 \\ 0 & \text{if} \quad w^i(p, q) = 1 \end{cases} .$$

The subspace dimensionality $\lambda(p, q)$ cannot be directly used as the subspace distance because points from parallel subspace clusters will have the same subspace preference vector. Thus, it has to be checked whether the subspace preference vectors of two points p and q are equal or one subspace preference vector is "included" in the other one. This can be done by computing the subspace preference vector $w(p,q)$ and checking whether $w(p,q)$ is equal to $w(p)$ or $w(q)$. If so, the distance between the points in the subspace spanned by $w(p,q)$ is determined. If this distance exceeds the threshold $2 \cdot \varepsilon$, the points belong to different, parallel clusters. The threshold ε, playing already a key role in the definition of the subspace dimensionality (cf. Definition 7.3) controls the degree of jitter of the subspace clusters.

Since $\lambda(p,q) \in \mathbb{N}$, there exist usually many tie situations when merging points/clusters during hierarchical clustering. These tie situations can be solved by considering the distance within a subspace cluster as a second criterion. Inside a subspace cluster, the points are then clustered in the corresponding subspace using the traditional OPTICS algorithm and, thus, the subspace clusters can exhibit arbitrary sizes, shapes, and densities.

Definition 7.6 (subspace distance).
Let w be an arbitrary preference vector. Then $S(w)$ is the subspace defined by w, i.e., $S(w) = \{A_i \mid A_i \in \mathcal{A} \wedge w^i = 1\}$. The inverse of w is denoted by $\bar{w}$, i.e.,

$$\bar{w}^i = \begin{cases} 0 & \text{if} \quad w^i = 1 \\ 1 & \text{if} \quad w^i = 0 \end{cases} \text{for } i = 1, \dots, d.$$

The subspace distance $\text{SD}\textsc{ist}$ *between two points $p, q \in \mathcal{D}$, denoted by $\text{SD}\textsc{ist}(p,q) = (d_1, d_2)$, is a pair consisting of the following two values*

$$\begin{aligned} d_1 &= \lambda(p,q) + \Delta(p,q) \\ d_2 &= \text{D}\textsc{ist}^{S(\bar{w}(p,q))}(p,q). \end{aligned}$$

$\Delta(p,q)$ is defined as

$$\Delta(p,q) = \begin{cases} 1 & \text{if } (w(p,q) = w_p \vee w(p,q) = w_q) \ \wedge \\ & \quad \text{D}\textsc{ist}^{S(w(p,q))}(p,q) > 2\varepsilon \\ 0 & \text{else} \end{cases}.$$

$\text{SD}\textsc{ist}(p,q) \leq \text{SD}\textsc{ist}(r,s)$ if one of the following conditions hold:

```
algorithm DiSH(Database $\mathcal{D}$, Real $\varepsilon$, Integer $\mu$)
    initialize empty subspace cluster order $co$;
    initialize empty priority queue $pq$ ordered by $\textsc{MinSReach}$;

    for each $p \in \mathcal{D}$ do
        compute $w_p$ w.r.t. parameters $\varepsilon$ and $\mu$;
        $p.\textsc{MinSReach} = \infty$;
        $pq$.insert($p$);
    end for

    while $pq \neq \emptyset$ do
        $p := pq$.next();
        $co$.add($p$);
        $r :=$ $\mu$-NN of $p$ w.r.t. $\textsc{SDist}$;

        for each $q \in pq$ do
            $sr_{new} := \max(\textsc{SDist}(p, r), \textsc{SDist}(p, q))$;

            if $sr_{new} < q.\textsc{MinSReach}$ then
                $q.\textsc{MinSReach} := sr_{new}$;
                $pq$.decrease($q$);
            end if

        end for

    end while

    return $co$;

end.
```

Figure 7.3: The DiSH algorithm.

1. $\textsc{SDist}(p, q).d_1 < \textsc{SDist}(r, s).d_1$

2. $\textsc{SDist}(p, q).d_1 = \textsc{SDist}(r, s).d_1$ *and* $\textsc{SDist}(p, q).d_2 \leq \textsc{SDist}(r, s).d_2$

As suggested in [ABKS99], a smoothing factor μ is introduced to avoid the Single-Link effect and to achieve robustness against noise points. The parameter μ represents the minimum number of points in a cluster and is equivalent to the parameter μ used to determine the best subspace for a point. Thus, instead of using the subspace distance $\textsc{SDist}(p, q)$ to measure the similarity of two points p and q, the *subspace reachability* $\textsc{SReach}_\mu(p, q)$ is used to compare these two points. The subspace reachability of a point q relative to a point p is defined as the maximum value of the subspace distance from p to its μ-nearest neighbor (w.r.t. the subspace distance $\textsc{SDist}$) and the subspace distance between p and q.

Definition 7.7 (subspace reachability).
For $\mu \in \mathbb{N}^+$, $\mu \leq |\mathcal{D}|$ let $r \in \mathcal{D}$ be the μ-nearest neighbor of $p \in \mathcal{D}$ w.r.t. the subspace distance $\textsc{SDist}$*. The* subspace reachability *of a point $q \in \mathcal{D}$ relative to point p w.r.t. μ is defined as*

$$\textsc{SReach}_\mu(p, q) = \max(\textsc{SDist}(p, r), \textsc{SDist}(p, q)).$$

The pseudocode of the DiSH algorithm can be seen in Figure 7.3. First, for each object its subspace preference vector is computed. In addition, for each point $p \in \mathcal{D}$ the minimum subspace reachability $p.\textsc{MinSReach}$ relative to any object processed before p is stored. Initially, $p.\textsc{MinSReach}$ is set to ∞ and p is added to a priority queue which stores all points according to $\textsc{MinSReach}$ in ascending order. DiSH selects in each step of the algorithm that point p having the minimum subspace reachability to any already processed point. The minimum subspace reachabilities of objects q which are still remaining in the priority queue are updated if their according values decrease. The resulting order of the points is called *subspace cluster order*. In a so-called subspace reachability diagram for each point, sorted according to the subspace cluster order along the x-axis, the subspace reachability value is plotted along the y-axis. The valleys in this diagram represent the subspace clusters.

7.3 Input Parameters

DiSH has two input parameters. The first parameter ε specifies the neighborhood of an object in an attribute. The ε-neighborhoods in each attribute are used to detect dimensions of low variance, i.e., high density. If the ε-neighborhood in one attribute contains at least the number of objects the second parameter μ specifies, the attribute is a so-called candidate attribute that might be used to build the subspace preference and the subspace dimensionality of a point (cf. Definition 7.2 and Definition 7.3). Thus, both input parameters are important for specifying the subspace preference vector and the subspace dimensionality of points. Additionally, the parameter ε specifies the amount of jitter that the projected clusters may exceed. In the experiments it turned out that DiSH is quite robust against the choice of ε, since ε is chosen between 0.1% and 0.5% of the attribute range. However, if subspace clusters with a lot of jitter are expected, ε can be increased accordingly.

The choice of parameter μ, specifying the minimum number of points in a cluster, is very intuitive. As mentioned above, it is needed for defining the subspace preference vector and the subspace dimensionality of a point In addition, it defines the density-based smoothing factor in order to avoid Single-Link effects during clustering. Again, the experimental evaluation shows that DiSH is quite robust against the choice of μ.

7.4 Runtime Complexity

Let n be the number of data points and d be the dimensionality of the data space. In the first loop, the subspace dimensionalities and preference vectors are precomputed which requires the determination of the best subspace for each object. Using the Apriori algorithm for determination of the best subspace, the complexity of the preprocessing step yields $O(n \cdot 2^d)$.

If the proposed heuristics is used (cf. Figure 7.1), the runtime of the preprocessing step results from the runtime of the determination of the candidate attributes and the complexity of the merging loop. The assignment of the candidate attributes requires the determination of the ε-neighborhood of each object in each dimension. Assuming a suitable index structure, this can be done in $O(\log n \cdot d)$ time for one object. If no such index is at hand, the time complexity of the determination of the candidate set is $O(n \cdot d)$ for one object. In each step of the merging loop, the intersection of the ε-neighborhoods in the already merged attributes with all ε-neighborhoods in the remaining candidate attributes has to be computed which needs $O(n \cdot d)$ time. Thus, the merging loop takes a runtime of $O(n \cdot d^2)$ for one object. Overall, for n objects the preprocessing step results in case of using the heuristics in $O(n^2 \cdot d^2)$.

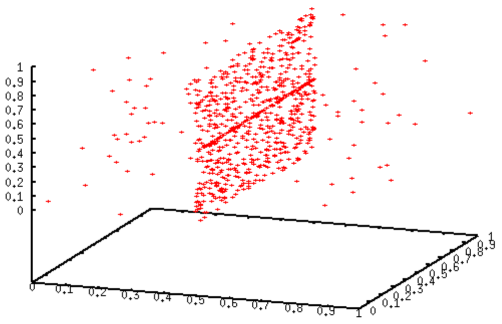
During the run of DiSH, for each point p of the data set the subspace reachability of all remaining points q in the priority queue to p has to be evaluated. This requires first the determination of the μ-nearest neighbor of p w.r.t. SDist which needs $O(n \cdot d)$ time for one object and $O(n^2 \cdot d)$ time for n objects. Then, the subspace reachability of q to p can be calculated in $O(d)$ time. Since this is done for all point pairs p and q, the complexity of the main loop of DiSH yields $O(n^2 \cdot d)$.

Overall, the complete runtime complexity of DiSH results in $O(n^2 \cdot d^2)$.

7.5 Visualizing Subspace Cluster Hierarchies

The reachability plot works well for the original OPTICS algorithm, but as output for DiSH it is not able to represent complex subspace hierarchies. Using the Euclidean distance function as similarity measure, the reachability plot reflects the density of the objects within the clusters. In contrast to the Euclidean distance, the new distance measure proposed for DiSH reflects the common subspace of two objects. Consequently, in that case the subspace reachability plot does not give the same information as for Euclidean distances.

Furthermore, the subspace reachability plot hides significant information. Consider the two data sets "DSA" and "DSB" depicted in Figures 7.4(a) and Figure 7.4(b). Data set "DSA" consists of a 1-dimensional line which is embedded in a 2-dimensional plane and some additional noise points. Data set "DSB" consists of two intersecting planes sharing a common intersection


(a) Data set “DSA”.

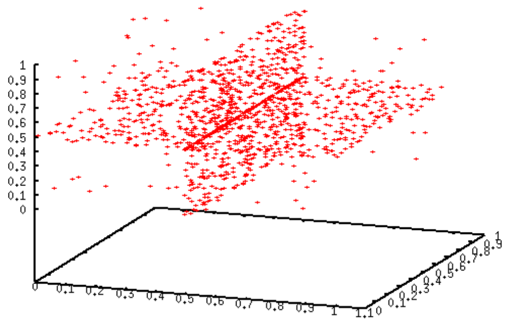
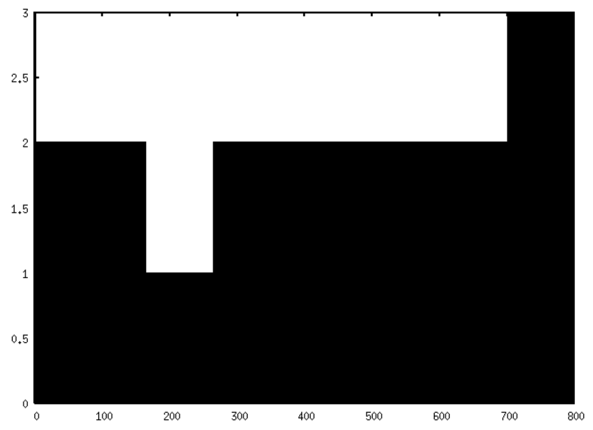

(b) Data set “DSB”.

Figure 7.4: Data sets with different hierarchies in 3-dimensional data.

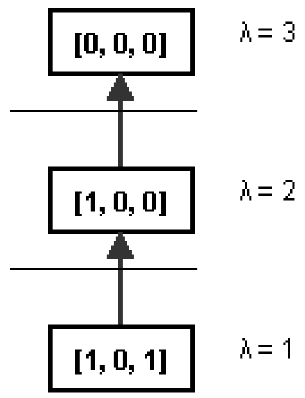
line. Exploring the subspace reachability plots of the two data sets, one can see that they look almost the same (cf. Figures 7.5(a) and 7.6(a)). Thus, taking only the subspace reachability plots into account, it is impossible to detect the obviously different kind of hierarchy of the second data set.

This phenomenon is due to the fact that the subspace dimensionality between an object of data set “DSB” belonging to the intersection line and an object belonging to the horizontal plane is the same as the subspace dimensionality between an object belonging to the intersection line and an object belonging to the vertical plane. Both pairs of objects span a 2-dimensional space: in the first case a horizontal plane, in the second case a vertical plane. For this reason, the subspace reachability plot of data set “DSB” shows no difference between objects of the horizontal and objects of the vertical plane, since both have a subspace reachability distance with a subspace dimensionality of two.

The limitation of the subspace reachability plot leads to the contribution of representing the relationships between subspace cluster hierarchies as a so-called *subspace clustering graph* so that the relationships between the subspace clusters can be explored at a glance. The subspace clustering graph displays a kind of hierarchy which should not be confused with a conventional (tree-like) cluster hierarchy as usually represented by dendrograms. The subspace clustering graph consists of nodes at several levels, where each level represents a subspace dimension. The top level represents the highest

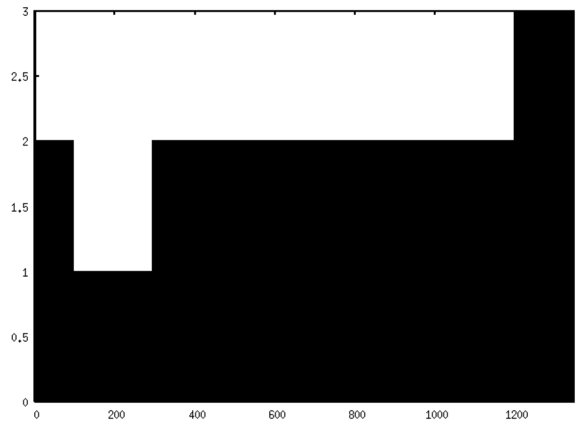


(a) Reachability plot.

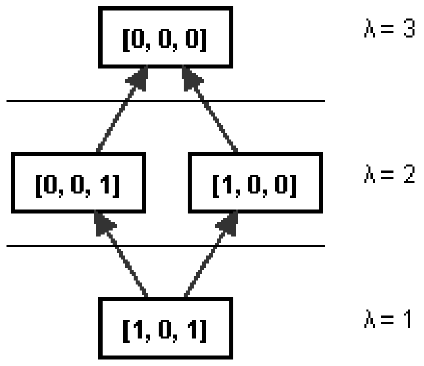


(b) Subspace clustering graph.

Figure 7.5: Results on data set "DSA".



(a) Reachability plot.



(b) Subspace clustering graph.

Figure 7.6: Results on data set "DSB".

subspace dimension which has the dimensionality of the data space. The subspace clustering graph consists of only one root node, representing all points that do not share a common subspace with any other point. These points are also called noise. Note that this is different to dendrograms where the root node represents the cluster of all objects. The nodes in the remaining levels represent clusters in the subspaces with the corresponding dimensionalities. They are labeled with the preference vector of the subspace cluster

```
function extractCluster(ClusterOrder $co$)
  initialize empty list of clusters $cl$;
  for each $p \in pq$ do
    $q :=$ $p$.predecessor;
    if $\nexists \mathcal{C} \in cl : w_{\mathcal{C}} = w(p,q) \wedge \textsc{Dist}^{S(w_{\mathcal{C}})}(p, \mathcal{C}.\text{centroid}) \leq 2 \cdot \varepsilon$ then
      initialize a new cluster $\mathcal{C}$ with $w_{\mathcal{C}} = w(p,q)$ and $\lambda_{\mathcal{C}} := \lambda(p,q)$;
      $cl$.add($\mathcal{C}$);
    end if
    $\mathcal{C}$.add($p$);
  end for
  return $cl$;
end.
```

Figure 7.7: The function to extract the subspace clusters from the cluster order.

they represent. For emphasizing the relationships between the subspace clusters, every subspace cluster is connected with its parents and its children. In contrast to tree representations, such as dendrograms, a graph representation allows multiple parents for a subspace cluster. This is necessary, since hierarchical subspace clusters can belong to more than one parent cluster. For instance, consider data set B, where the objects of the intersection line are embedded in the horizontal plane as well as in the vertical plane, i.e., the subspace cluster forming the intersection line belongs to two parents in the hierarchy. The subspace clustering graphs of the two data sets “DSA” and “DSB” are depicted in Figures 7.5(b) and 7.6(b). The line of data set “DSA” is represented by the subspace cluster with the subspace preference vector [1,0,1]. This subspace cluster is a child of subspace cluster [1,0,0], representing the plane in data set “DSA” (cf. Figure 7.5(b)). The more complex hierarchy of data set “DSB” is represented in Figure 7.6(b), where the subspace cluster [1,0,1] of the intersection line belongs to two parent clusters, the subspace cluster of the horizontal plane [0,0,1] and the subspace cluster

```
procedure buildHierarchy(ClusterList $cl$)
  $\lambda_{max} := d$; // $d$ = dimensionality of data space
  for each $\mathcal{C}_i \in cl$ do
    for each $\mathcal{C}_j \in cl$ with $\lambda_{\mathcal{C}_i} < \lambda_{\mathcal{C}_j}$ do
      if $\lambda_{\mathcal{C}_j} = \lambda_{max} \wedge \mathcal{C}_i$.parents=$\emptyset$ then
        $\mathcal{C}_i$.addParent($\mathcal{C}_j$);
      else
        if $\lambda(\mathcal{C}_i, \mathcal{C}_j) + \Delta(\mathcal{C}_i, \mathcal{C}_j) = \lambda_{\mathcal{C}_j} \wedge$
          ($\mathcal{C}_i$.parents=$\emptyset \vee \neg$ isParent($\mathcal{C}_j$, $\mathcal{C}_i$.parents))
        then
          $\mathcal{C}_i$.addParent($\mathcal{C}_j$);
        end if
      end if
    end for
  end for
end.
```

Figure 7.8: The procedure to build the hierarchy of subspace clusters.

of the vertical plane [1,0,0].

In contrast to dendrograms, objects are not placed in singleton clusters at the leaf level, but are assigned to the lowest dimensional subspace cluster they fit in within the graph. Similar to dendrograms, an inner node N of the subspace graph represents the subspace cluster of all points that are assigned to N and of all points assigned to its child clusters.

To build the subspace clustering graph, in a first step all subspace clusters are extracted from the subspace cluster order. A subspace cluster $\mathcal{C}$ consists of a set of objects belonging to $\mathcal{C}$, the preference vector $w_{\mathcal{C}}$ of $\mathcal{C}$ indicating the subspace in which the objects of $\mathcal{C}$ are embedded, and the subspace dimensionality $\lambda_{\mathcal{C}}$ of $\mathcal{C}$. Note that parallel subspace clusters share the same

```
function isParent(Cluster $\mathcal{P}$, ClusterList $cl$)
    for each $\mathcal{C} \in cl$ do
        if $\lambda(\mathcal{P},\mathcal{C}) + \Delta(\mathcal{P},\mathcal{C}) = \lambda_{\mathcal{P}}$ then
            return true;
        end if
    end for
    return false;
end.
```

Figure 7.9: The function to check whether a cluster is a parent of one of the clusters in a list.

subspace preference vector and the same subspace dimensionality. For each object p in the subspace cluster order the appropriate subspace cluster $\mathcal{C}$ has to be found. A subspace cluster $\mathcal{C}$ accommodates object p if the subspace preference vector $w_{\mathcal{C}}$ of $\mathcal{C}$ is equal to the subspace preference vector $w(p,q)$ between object p and its predecessor q. Additionally, since parallel subspace clusters share the same subspace preference vector, the distance between the centroid of the subspace cluster $\mathcal{C}$ and object p in subspace $S(w_{\mathcal{C}})$ has to be less than or equal to $2 \cdot \varepsilon$. If no such subspace cluster exists, a new subspace cluster $\mathcal{C}$ with subspace preference vector $w_{\mathcal{C}} = w(p,q)$ and subspace dimensionality $\lambda_{\mathcal{C}} = \lambda(p,q)$ is created to accommodate object p. The complete method to extract the subspace clusters from the subspace cluster order can be seen in Figure 7.7.

After the subspace clusters have been derived from the subspace cluster order, the second step builds the subspace cluster hierarchy. For each subspace cluster $\mathcal{C}_i$ it has to be checked if $\mathcal{C}_i$ is part of one or more higher dimensional subspace clusters $\mathcal{C}_j$, whereas each subspace cluster is at least the child of the noise cluster. In particular, the following steps are applied for each subspace cluster $\mathcal{C}_i$ and each subspace cluster $\mathcal{C}_j$ with $\lambda_{\mathcal{C}_i} < \lambda_{\mathcal{C}_j}$: If no parents have already been assigned to subspace cluster $\mathcal{C}_i$ and subspace cluster $\mathcal{C}_j$ is the noise cluster, i.e., the subspace dimensionality $\lambda_{\mathcal{C}_j}$ of subspace cluster $\mathcal{C}_j$ equals the dimensionality of the feature space, subspace cluster $\mathcal{C}_j$ will become the (only) parent of subspace cluster $\mathcal{C}_i$.

Otherwise it is checked if both subspace clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ together form a $\lambda_{\mathcal{C}_j}$-dimensional subspace cluster. For this purpose, the subspace dimensionality $\lambda(\mathcal{C}_i, \mathcal{C}_j)$ and the value of $\Delta(\mathcal{C}_i, \mathcal{C}_j)$ has to be computed. The subspace dimensionality $\lambda(\mathcal{C}_i, \mathcal{C}_j)$ of the common subspace of $\mathcal{C}_i$ and $\mathcal{C}_j$ is the number of zero-values in the subspace preference vector $w(\mathcal{C}_i, \mathcal{C}_j)$. The subspace preference vector $w(\mathcal{C}_i, \mathcal{C}_j)$ of subspace clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ can be computed analogously to the subspace preference vector of a pair of points (cf. Definition 7.4), i.e., $w(\mathcal{C}_i, \mathcal{C}_j)$ is the attribute wise logical AND-conjunction of the subspace preference vector $w_{\mathcal{C}_i}$ of subspace cluster $\mathcal{C}_i$ and the subspace preference vector $w_{\mathcal{C}_j}$ of subspace cluster $\mathcal{C}_j$. The value of $\Delta(\mathcal{C}_i, \mathcal{C}_j)$ is determined analogously to Definition 7.6:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \begin{cases} 1 & \text{if } (w(\mathcal{C}_i, \mathcal{C}_j) = w(\mathcal{C}_i) \vee w(\mathcal{C}_i, \mathcal{C}_j) = w(\mathcal{C}_j)) \;\; \wedge \\ & \qquad \textsc{Dist}^{S(w(\mathcal{C}_i, \mathcal{C}_j))}(\mathcal{C}_i.\text{centroid}, \mathcal{C}_j.\text{centroid}) > 2\varepsilon \\ 0 & \text{else} \end{cases}.$$

If both subspace clusters together form a $\lambda_{\mathcal{C}_j}$-dimensional subspace cluster, subspace cluster $\mathcal{C}_j$ will become a parent of subspace cluster $\mathcal{C}_i$, provided that one of the following conditions holds: Either $\mathcal{C}_i$ has no parents so far or $\mathcal{C}_j$ is no parent cluster of the already assigned parents of subspace cluster $\mathcal{C}_i$, because in that case the relationship between $\mathcal{C}_j$ and $\mathcal{C}_j$ is that of a grandparent. The methods used to build the subspace hierarchy from the subspace clusters are depicted in Figure 7.8 and Figure 7.9.

7.6 Experimental Evaluation

All experiments have been performed on a workstation with a $2 \cdot 64$-bit 2.6 GHz CPU and 16 GB main memory. All evaluated methods have been implemented in Java. In all experiments, the input parameters of all methods have been optimized in terms of quality and the best results have been reported in order to achieve a fair comparison.

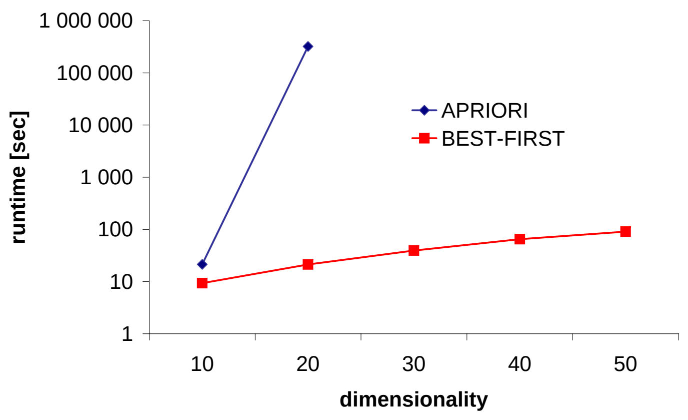


Figure 7.10: Runtime of the preprocessing step of DiSH w.r.t. the strategy for preference vector computation.

7.6.1 Strategy Used for the Preprocessing Step

In a first experiment, the runtime of the preprocessing step of DiSH w.r.t. the strategies used for the determination of the best subspace has been evaluated. The experiment was run on data sets with 1,000 objects with varying dimensionality of $d = 10, \ldots, 50$. For each data set, the objects were equally distributed over 10 subspace clusters, where the attribute values are in the range of 0.0 to 1.0. The parameters for DiSH were set to $\mu = 50$ and $\varepsilon = 0.1\%$ of the attribute range, i.e., $\varepsilon = 0.001$. The results are shown in Figure 7.10. The strategy using the Apriori-algorithm [AS94] is denoted with “APRIORI”, the heuristics using the best-first search is denoted with “BEST-FIRST”. As it can be seen, the heuristics using best-first search outperforms the strategy using the Apriori-algorithm in terms of runtime significantly (note the logarithmic scale). Since both strategies have equal F-measures of 100% in this experiment, in all further experiments the heuristics has been used to compute the preference vectors rather than the Apriori-based approach.

7.6.2 Effectivity

Synthetic Data Sets. DiSH has been evaluated on several synthetic data sets. The results on three data sets named “DS1”, “DS2” and “DS3” are

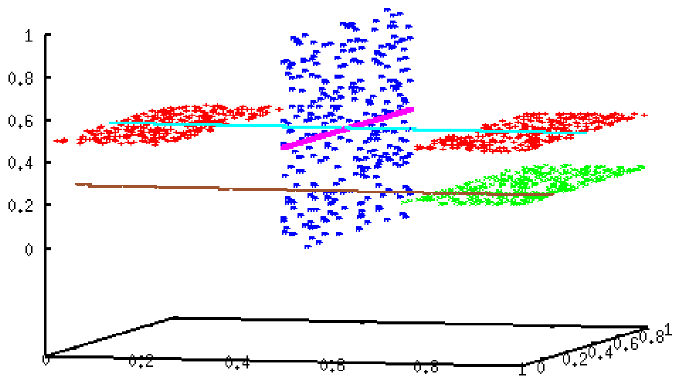
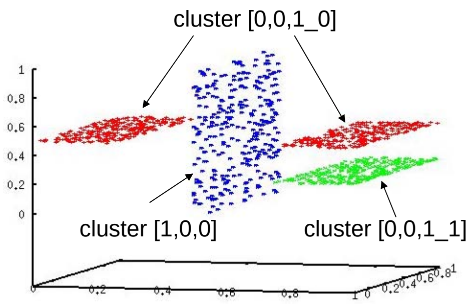


Figure 7.11: Data set "DS1" (noise points are not depicted).

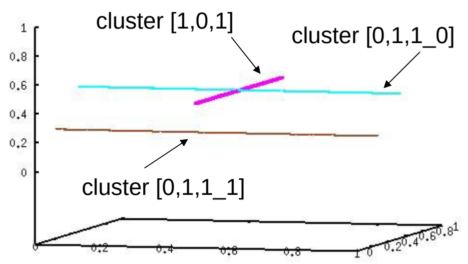
shown. The synthetic data sets contain 3-, 5- and 16-dimensional objects grouped in complex hierarchies of subspace clusters with several multiple inclusions and additional noise points. The attribute values of all synthetic data sets are in the range of 0.0 to 1.0.

Data set "DS1" (cf. Figure 7.11) contains 3-dimensional points grouped in a complex hierarchy of three 1-dimensional and three 2-dimensional subspace clusters with several multiple inclusions and additional noise points (not depicted). The results of DiSH applied to "DS1" are shown in Figure 7.12. The parameters of DiSH were set to $\varepsilon = 0.001$ and $\mu = 20$. In the upper left Figure 7.12(a) the three 2-dimensional subspace clusters found by DiSH are marked with different colors. The upper right Figure 7.12(b) shows the three 1-dimensional subspace clusters found by DiSH. In the lower Figure 7.12(c) the resulting subspace clustering graph is displayed. As it can be seen, the complete hierarchical subspace clustering structure can be obtained from the subspace clustering graph. In particular, the complex nested subspace clustering structure can be seen at a glance.

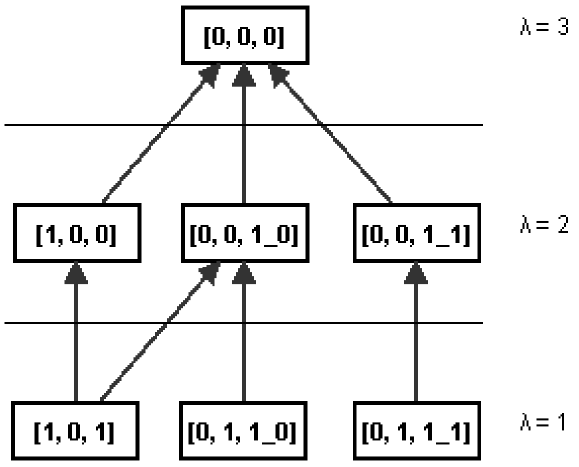
For comparison, HiSC, PreDeCon and PROCLUS have also been applied to data set "DS1". The resulting subspace distance diagram of HiSC on "DS1" (with parameter setting $k = 20$ and $\alpha = 0.001$) is shown in Figure 7.13. Though, HiSC detects the simple hierarchical relationship that the 1-dimensional subspace clusters are embedded in the 2-dimensional subspace clusters, it fails completely in discovering the complex relationships of mul-



(a) 2-dimensional subspace clusters.



(b) 1-dimensional subspace clusters.



(c) Subspace clustering graph.

Figure 7.12: Results of DiSH on "DS1".

tiple inclusion as DiSH does (cf. also the discussion of Figure 7.4 in Section 7.5).

PreDeCon can either detect the 1-dimensional subspace clusters or the 2-dimensional subspace clusters, but not both in one single run. The results of PreDeCon with different settings for parameter λ which determines the subspace dimensionality of the clusters to be found, are depicted in Figure 7.14. The left Figure 7.14(a) shows the two 1-dimensional subspace clusters found by PreDeCon with parameter setting $\varepsilon = 0.05, \mu = 20, \lambda = 1, \delta = 0.0001$. Although trying different settings, PreDeCon was not able to separate the two intersecting lines correctly, i.e. PreDeCon always assigned them to one cluster. Furthermore, in this run, the 2-dimensional planes were classified as

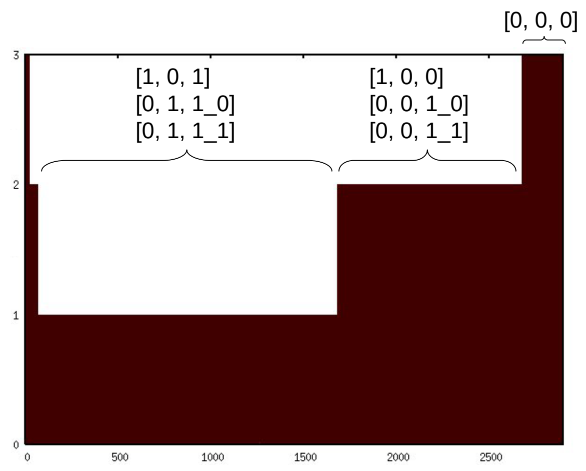
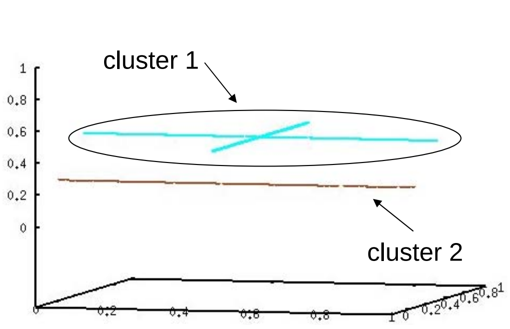


Figure 7.13: Subspace distance diagram of HiSC on "DS1".

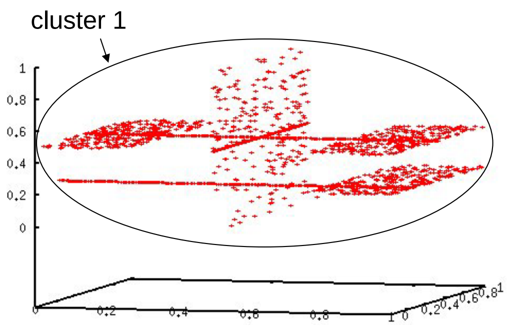
noise. In the right Figure 7.14(b) the 2-dimensional subspace cluster detected by PreDeCon with parameter setting $\varepsilon = 0.2, \mu = 20, \lambda = 2, \delta = 0.0001$ is depicted. This cluster includes the three 1-dimensional subspace clusters and the two 2-dimensional subspace clusters. As it can be seen in Figure 7.15, PROCLUS ($k = 6, l = 3$) failed completely in finding the subspace clusters in "DS1". Overall, neither PreDeCon nor PROCLUS were able to detect the hierarchies and all subspace clusters in data set "DS1".

The synthetic data set "DS2" is a 5-dimensional data set containing ten subspace clusters of different dimensionality and noise with several multiple inclusions: one subspace cluster is embedded in a 4-dimensional subspace, four subspace clusters are 3-dimensional, three subspace clusters are 2-dimensional and two subspace clusters are 1-dimensional subspace clusters. The resulting subspace clustering graph produced by DiSH (with parameter setting $\varepsilon = 0.001$ and $\mu = 20$) is shown in Figure 7.16 and exhibits all ten subspace clusters of considerably different dimensionality correctly. The relationships between the different subspace clusters have all been accurately determined.

Similar observations can be made when evaluating the subspace clustering graph obtained by DiSH on data set "DS3" (cf. Figure 7.17). The 16-dimensional data set "DS3" contains noise points, one 13 dimensional,



(a) Result of PreDeCon with $\lambda = 1$.



(b) Result of PreDeCon with $\lambda = 2$.

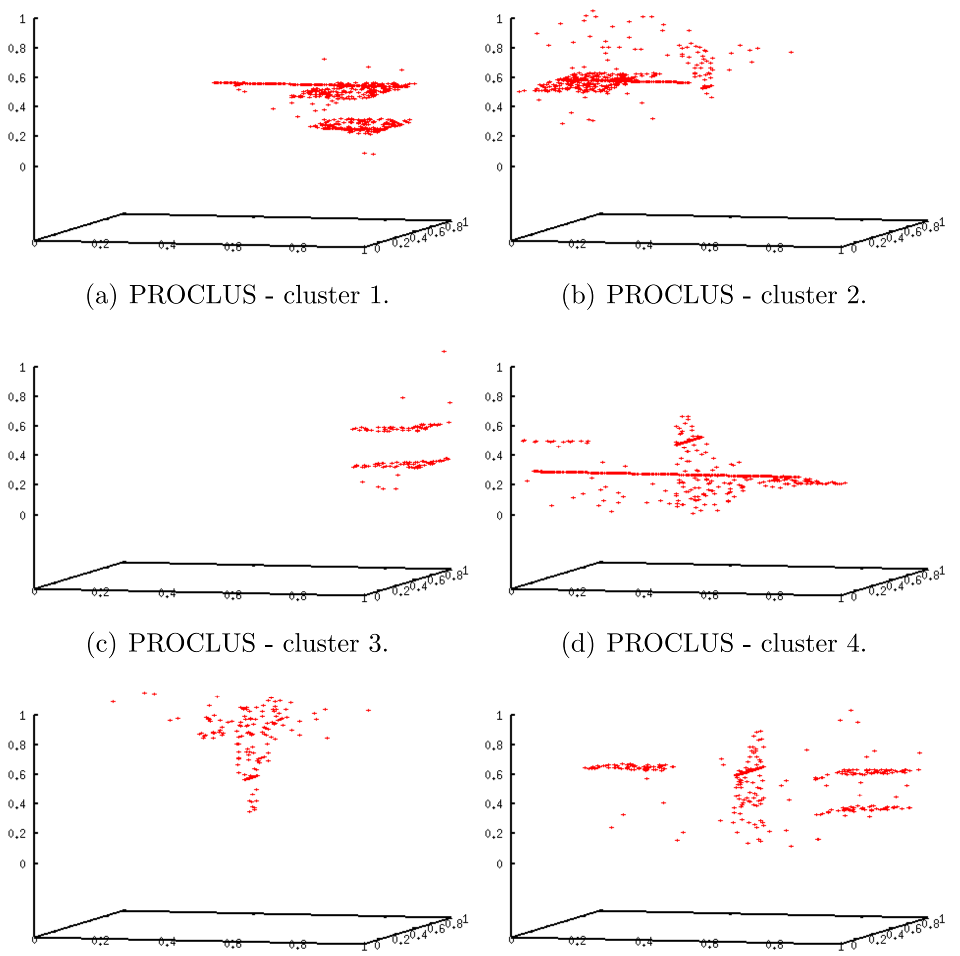
Figure 7.14: Results of PreDeCon with different λ-parameter settings on "DS1".

one 11 dimensional, one 9 dimensional, one 7 dimensional subspace cluster, and two 6 dimensional subspace clusters. Again, DiSH found all six subspace clusters and the relationships between the different subspace clusters correctly.

Again, HiSC, PreDeCon and PROCLUS have also been applied to the two data sets "DS2" and "DS3". Although trying several parameter settings, all three algorithms failed to detect the hierarchies or the subspace clusters of significantly different dimensionality.

Real-world Data Sets. In addition to the synthetic data sets, the effectivity of DiSH has been evaluated by using several real-world data sets. First, DiSH has been applied on the "Wages" data set[1], consisting of 534 11-dimensional observations from the 1985 Current Population Survey. Since most of the attributes are not numeric, only 4 dimensions (YE=years of education, W=wage, A=age, and YW=years of work experience) have been used for clustering. The resulting subspace cluster hierarchy (using $\varepsilon = 0.001$, $\mu = 9$) is visualized in Figure 7.18. The nine parallel subspace clusters, having a subspace dimensionality of $\lambda = 3$, consist of data of people having equal years of education, e.g., in subspace cluster $[1, 0, 0, 0\_0]$ YE=17 and

[1]http://lib.stat.cmu.edu/datasets/CPS_85_Wages



(a) PROCLUS - cluster 1.
(b) PROCLUS - cluster 2.
(c) PROCLUS - cluster 3.
(d) PROCLUS - cluster 4.
(e) PROCLUS - cluster 5.
(f) PROCLUS - cluster 6.

Figure 7.15: Results of PROCLUS on "DS1".

in subspace cluster [1, 0, 0, 0_5] YE=12. The two subspace clusters labeled with [1, 1, 0, 0_0] and [1, 1, 0, 0_1] in the 2-dimensional subspace are children of subspace cluster [1, 0, 0, 0_5] and have (in addition to equal years of education, YE=12) equal wages values (W=7.5 and W=5, respectively). The 1-dimensional subspace cluster [1, 0, 1, 1] is a child of [1, 1, 0, 0_0] and has the following properties: YE=12, A=26, and YW=8.

Then, DiSH has been applied to the original Wisconsin Breast Cancer

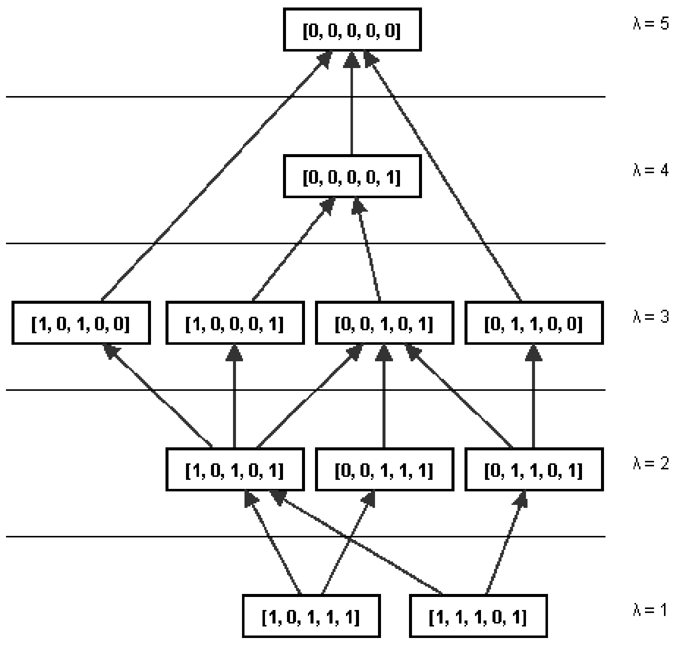


Figure 7.16: Results of DiSH on synthetic data set “DS2”.

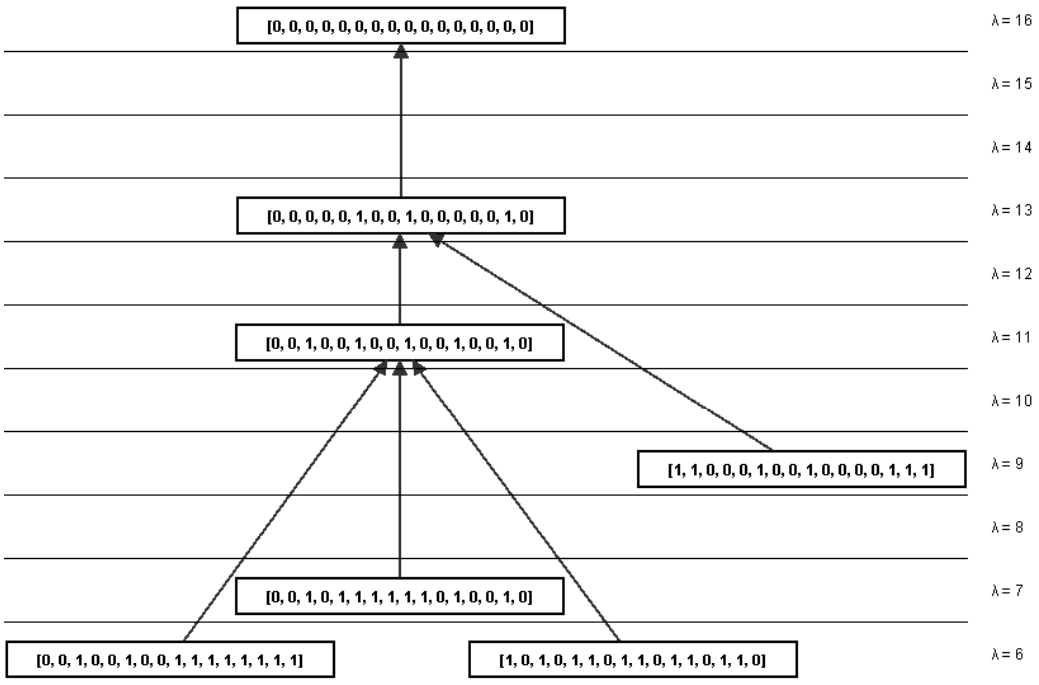


Figure 7.17: Results of DiSH on synthetic data set “DS3”.

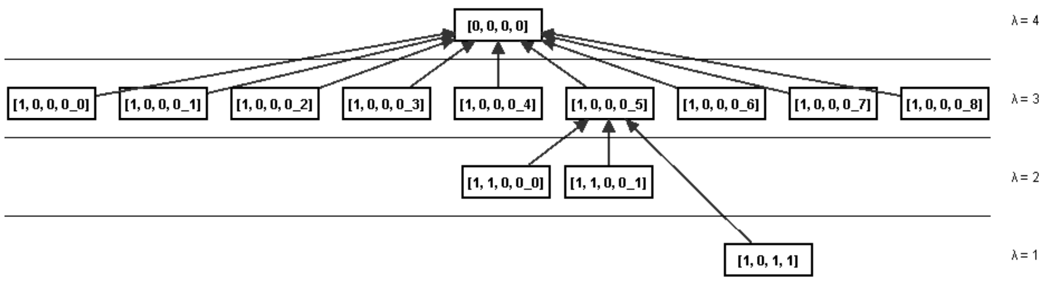


Figure 7.18: Result of DiSH on "Wages" data.

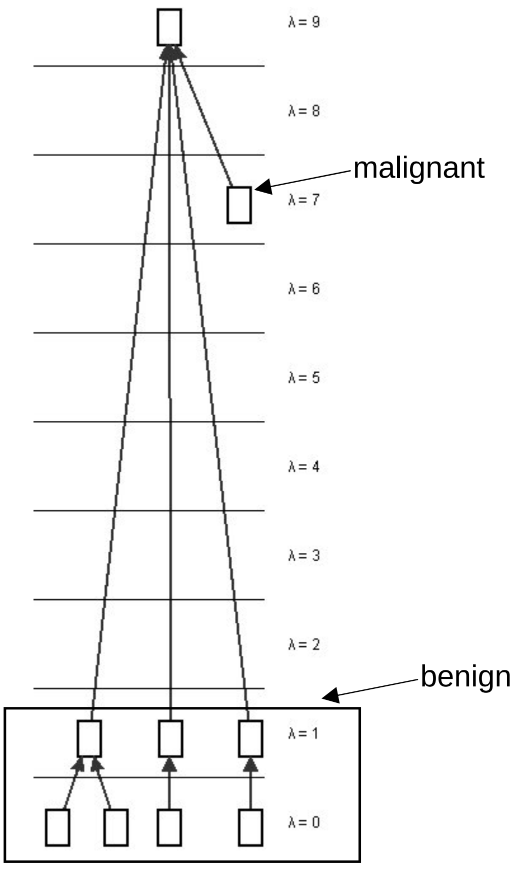


Figure 7.19: Result of DiSH on "Breast Cancer" data.

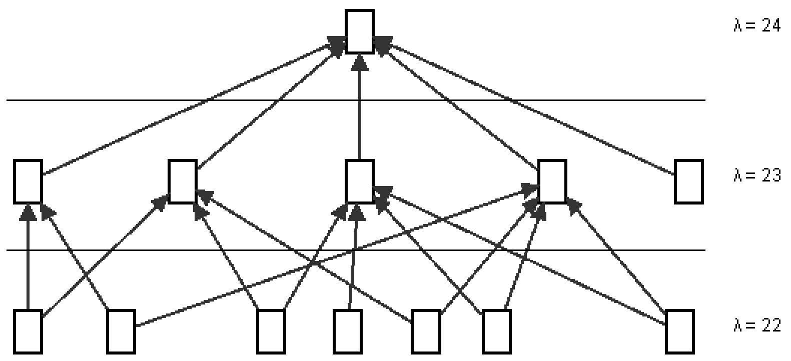


Figure 7.20: Result of DiSH on "Gene Expression" data.

Database from the UCI ML Archive[2]. This data set, in the following named "Breast Cancer", contains 9 measurements for each of 683 patients suffering either from a malignant or from a benign type of breast cancer. Figure 7.19 depicts the results of DiSH applied to "Breast Cancer" using $\varepsilon = 0.01$ and $\mu = 15$. As it can be seen, the resulting hierarchy contains several low dimensional subspace clusters and one 7-dimensional subspace cluster. It is worth mentioning that the reported subspace clusters are pure, i.e., they only contain patients of either malignant or benign type. In particular, the seven low-dimensional subspace clusters only contain benign patients, whereas the 7-dimensional subspace cluster only contains malignant patients. Some patients from both classes could not be separated and were labeled as noise.

Last but not least, DiSH has been applied to the "Gene Expression" data set of [SSZ+98], consisting of the expression level of approximately 4,000 yeast genes measured at 24 time spots during mitosis. The result of DiSH (using $\varepsilon = 0.01$, $\mu = 100$) on the "Gene Expression" data is shown in Figure 7.20. Again, DiSH found several subspace clusters of different subspace dimensionalities with multiple inclusions.

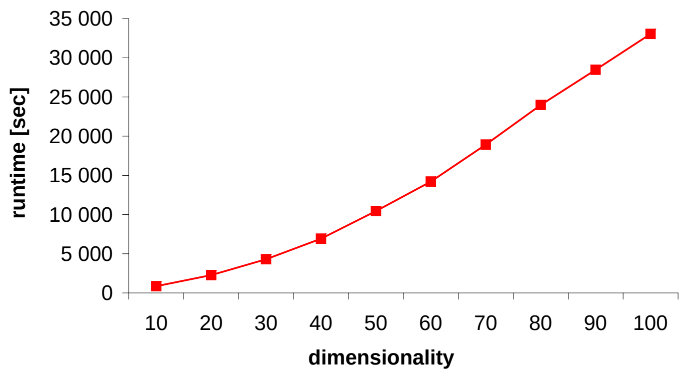


Figure 7.21: Scalability of DiSH w.r.t. the dimensionality.

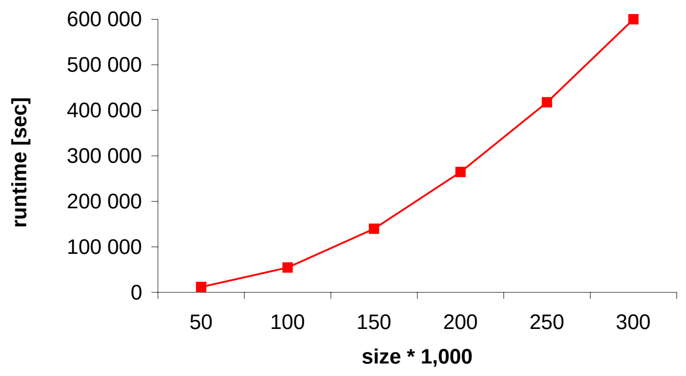


Figure 7.22: Scalability of DiSH w.r.t. the size.

7.6.3 Scalability

The scalability of DiSH w.r.t. the dimensionality of the data set is depicted in Figure 7.21. The experiments were obtained by using 10 data sets of 10,000 objects with varying dimensionality of $d = 10, 20, 30, \ldots, d_{max} = 100$. For each data set, the objects were equally distributed over 10 subspace clusters, where the single attributes have values in the range of $[0.0, 1.0]$. The result shows a quadratic increase of runtime when increasing the dimensionality of the data set. The parameters for DiSH were set to $\mu = 3 \cdot d_{max} = 300$ and $\varepsilon = 0.1\%$ of the attribute range, i.e., $\varepsilon = 0.001$. This experiment confirms the theoretically determined runtime complexity of $O(n^2 \cdot d^2)$ (cf. Section

[2]http://www.ics.uci.edu/~mlearn/MLSummary.html

7.4).

A similar observation can be made when evaluating the scalability of DiSH w.r.t. the data set size (cf. Figure 7.22). The experiment was run on a set of 6 10-dimensional synthetic data sets with increasing number of objects ranging from 50,000 to 300,000. The objects are equally distributed over nine subspace clusters of subspace dimensionality $\lambda = 1, \ldots, 9$ and noise, where the attribute values are in the range of 0.0 to 1.0. The parameters for DiSH were set to $\mu = 3 \cdot d = 15$ and $\varepsilon = 0.1\%$ of the attribute range, i.e., $\varepsilon = 0.001$. As it can be seen, DiSH scales quadratic w.r.t. the number of tuples in the data set. Again, this experiments confirms the theoretically determined runtime complexity of $O(n^2 \cdot d^2)$.

Part III

Hierarchical Subspace Clustering in Arbitrarily Oriented Subspaces

Chapter 8

Introduction

In high-dimensional data, meaningful clusters are usually based only on a subset of all dimensions. Axis-parallel subspace clustering (or projected clustering) as described in Part II is a well-known approach to find λ-dimensional axis-parallel subspaces in a d-dimensional data space ($\lambda < d$) where certain sets of points cluster well. Since the number of possible subspaces is exponential in the dimensionality of the data space, all existing approaches are based on some heuristics and therefore suffer from certain drawbacks.

An even more challenging problem is to find clusters in arbitrarily oriented subspaces, also called *generalized subspace clustering*, where the number of possible subspaces is even infinite. Such subspace clusters appear as sets of points located near a common hyperplane of arbitrary dimension λ in a d-dimensional data space. Since these hyperplanes correspond to linear dependencies among several attributes and, thus, the corresponding attributes are correlated, the concept of knowledge discovery in databases addressing this problem is also known as *correlation clustering*. For a detailed discussion on existing algorithms for correlation clustering please refer to Chapter 9.

Correlation clustering groups the data sets into subsets called correlation clusters such that the objects in the same correlation cluster are all associated to a common hyperplane of arbitrary dimensionality λ. A correlation cluster associated to a λ-dimensional hyperplane is referred to as a λ-*dimensional* *correlation cluster*. The dimensionality of a hyperplane associated to a correlation cluster is called the *correlation dimensionality*. Intuitively, the problem of correlation clustering can be formalized as follows:

Given a set of n points from a d-dimensional vector space and a natural number μ. Find a set of λ-dimensional hyperplanes in $\mathbb{R}^d$ ($\lambda < d$) of minimum dimensionality to each of which at least μ points can be assigned such that the distance between the hyperplanes and the associated points is minimized.

For example, correlation clustering can be successfully applied in recommendation systems. In target marketing, it is important to find homogeneous groups of users with similar ratings in subsets of the attributes. In addition, it is interesting to find groups of users with correlated affinities. This knowledge can help companies to predict customer behavior and thus develop future marketing plans. A second application of correlation clustering is metabolic screening. The collected data usually contain the concentrations of certain metabolites in the blood of thousands of patients. In such data sets, it is important to find homogeneous groups of patients with correlated metabolite concentrations indicating a common metabolic disease. This is an important step towards understanding metabolic or genetic disorders and designing individual drugs. Another prominent application for correlation clustering is the analysis of gene expression data. Gene expression data contain the expression levels of thousands of genes, indicating how *active* the genes are, according to a set of samples. A common task is to find clusters of co-regulated genes, i.e., clusters of genes that share a common linear dependency within a set of their features.

In [BKKZ04] it was demonstrated that none of the two obvious simple approaches is sufficient to find correlation clusters. The first obvious approach is to apply PCA (Principal Component Analysis) for dimensionality reduction and to perform a cluster analysis afterwards. This approach will fail if different correlation clusters are associated to hyperplanes of different dimensionality or direction. The other obvious approach, first searching for clusters and after that applying a dimensionality reduction technique on top of the detected clusters, fails if some of the correlation clusters touch or cross

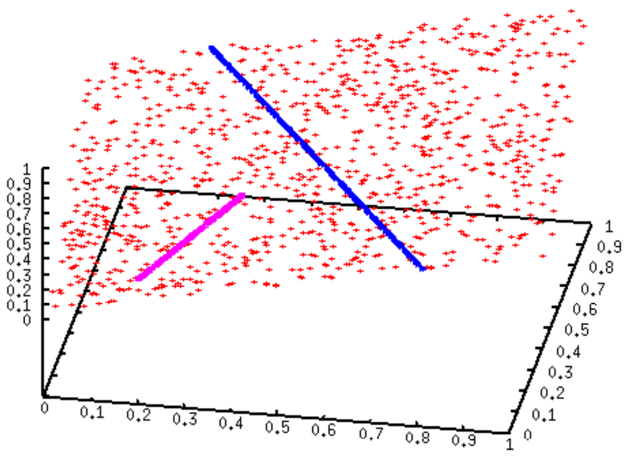


Figure 8.1: Hierarchies of correlation clusters.

each other. In this case, the clustering algorithm cannot separate the correlation clusters correctly and the joint correlation cluster is not amenable to dimensionality reduction any more. Therefore, algorithms for correlation clustering have to integrate the concepts of clustering and correlation detection in a more sophisticated way. The algorithm ORCLUS [AY00], for instance, integrates PCA into k-means clustering and the algorithm 4C [BKKZ04] integrates PCA into the density based clustering algorithm DBSCAN [EKSX96]. Both algorithms decompose a data set into subsets of points, each subset being associated to a specific λ-dimensional hyperplane. Since both methods use the correlation dimensionality λ as a global parameter, i.e., the correlation dimensionality of the resulting correlation clusters must be determined by the user, they are both not able to find all correlation clusters of different dimensionality during one single run.

Moreover, searching clusters of different dimensionality is basically a hierarchical problem, because several correlation clusters of low dimensionality may together form a larger correlation cluster of higher dimensionality, and so on. As a simple illustration consider the data set depicted in Figure 8.1. Two lines, i.e., two 1-dimensional correlation clusters, are embedded within a plane which forms a 2-dimensional correlation cluster. In order to detect the lines, a search for 1-dimensional correlation clusters has to be performed, whereas in order to detect the plane, a search for 2-dimensional

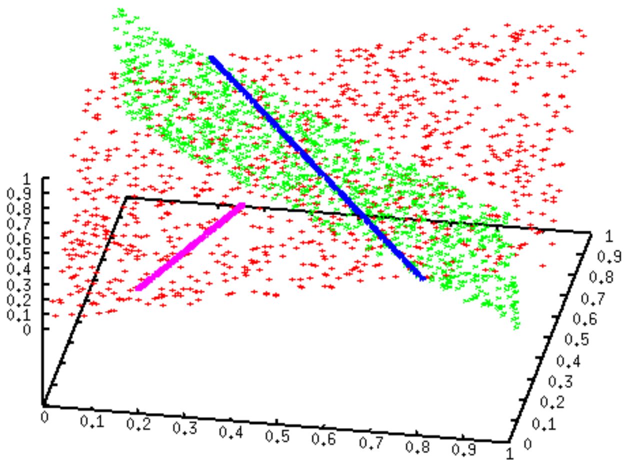


Figure 8.2: Hierarchies of correlation clusters with multiple inheritance.

correlation clusters has to be started. The resulting hierarchy consists of two 1-dimensional clusters as leaf nodes of the hierarchy tree, both having the 2-dimensional correlation cluster as parent node.

None of the existing algorithms (cf. Chapter 9 for details) is able to detect hierarchies of correlation clusters. Therefore, in this Part three new hierarchical approaches for correlation clustering are proposed. Chapter 10 introduces the algorithm HiCO (Hierarchical Correlation Ordering), a new algorithm for searching simultaneously for correlation clusters of arbitrary dimensionality and detecting hierarchies of correlation clusters. The cluster hierarchy is visualized by using so-called correlation reachability plots, which are tree-based representations of the correlation cluster hierarchy. Additionally, HiCO overcomes another drawback of the existing non-hierarchical correlation clustering methods like 4C and ORCLUS: HiCO does not require the user to define critical parameters that limit the quality of clustering such as a density threshold or the number of clusters in advance.

However, it may not always be appropriate to reflect the hierarchy of correlation clusters as a tree. A correlation cluster may be embedded in several correlation clusters of higher dimensionality, resulting in a hierarchy with multiple inclusions (similar to the concept of "multiple inheritance" in software engineering). As an example consider the data set depicted in Figure 8.2: One of the 1-dimensional correlation clusters is the intersection of two 2-dimensional correlation clusters, i.e., it is embedded in two clusters of higher dimensionality. Those multiple inclusions can only be represented by a graph-based visualization approach. In Chapter 11 a new algorithm called ERiC (Exploring complex hierarchical Relationships among Correlation clusters) is proposed. ERiC is able to uncover complex hierarchical relationships of correlation clusters in high-dimensional data sets including not only single inclusions (like HiCO) but also multiple inclusions. In addition, ERiC provides a clear visualization of these complex relationships by means of a graph-based representation.

Chapter 9

Related Work

In the following related work on correlation clustering is reviewed and discussed. Please note that none of the existing approaches to correlation clustering can detect hierarchies of correlation clusters, i.e., lower-dimensional correlations within a common higher-dimensional correlation.

Pattern Based Clustering methods [YWWY02, WWYY02, PZC+03, LW03] aim at grouping objects that exhibit a similar trend in a subset of attributes into clusters rather than objects with low distance. This problem is also known as co-clustering or biclustering [Har72], as recently proposed e.g. for gene expression data analysis [CC00]. A detailed comparison of biclustering methods is presented in [MO04]. In contrast to correlation clustering, pattern-based clustering is limited to a special form of correlation where all attributes are positively correlated. It does not capture negative correlations nor correlations where one attribute is determined by two or more other attributes. Thus, biclustering or pattern-based clustering could be regarded as a special case of correlation clustering, as more extensively discussed in [BKKZ04].

EM (Expectation Maximization) [DLR77] is one of the first clustering algorithms that can detect correlation clusters. The EM algorithm tries to model the data distribution of a data set using a mixture of non-axis parallel Gaussian distributions. The algorithm starts with an initial clustering and alternates between performing an expectation (E) step and a maximization (M) step. The expectation step determines for each object based on an assumed probability distribution the likelihood of the object belonging to each cluster. The maximization step recomputes the clusters by maximizing the expected likelihood found on the E step. The new clusters found on the M step are then used to begin another E step, and the process is repeated until the increase in likelihood becomes negligible. Although the EM-algorithm is guaranteed to converge to a maximum, this maximum may not necessarily be the same as the global maximum. However, the EM clustering cannot distinguish between correlation clusters and full-dimensional clusters without any correlation. In addition, it favors clusters of spherical shape and requires the user to specify the number of clusters in advance. As a main drawback, the EM clustering is very sensitive to noise and the quality of the clustering result depends significantly on the chosen probability distribution.

ORCLUS (arbitrarily ORiented projected CLUSter generation) [AY00] is an extended version of the algorithm PROCLUS [APW+99] that finds correlation clusters in arbitrarily oriented subspaces. The user has to define two input parameters, the number k of clusters to be found and the approximate dimensionality l of the subspace containing a cluster. A generalized projected Cluster is formed by a subspace E which is defined by a set of orthonormal basis vectors and a cluster C consisting of a set of data points, such that the points in C are closely clustered in E. ORCLUS iteratively performs the following three steps until k clusters are found: cluster assignment, subspace determination and merging. In the assignment step the data set is partitioned into k_c current clusters by assigning each data point to its nearest seed using the projected distance in the current subspace E_i. Then, each seed is replaced by the centroid of the newly created cluster. Subspace determination finds the subspace E_i of dimensionality l_c for each current cluster C_i by calculating the covariance matrix of C_i and selecting the l_c eigenvectors having the smallest eigenvalues to form the basis of E_i. In each iteration step the dimensionality l_c of the current subspace is reduced by fraction $\beta < 1$. Clusters that are near to each other and have similar directions are merged during the merge phase, while the number k_c of current clusters is reduced by a factor $\alpha < 1$. The algorithm terminates when the number k_c of current clusters is equal to the input parameter k, which denotes the number of clusters to be found. If the user's guess of k does not correspond to the actual number of clusters, the results of ORCLUS deteriorate considerably. A second problematic parameter of ORCLUS is the dimensionality l of the correlation clusters ORCLUS is desired to uncover. ORCLUS usually has problems with correlation clusters of very different dimensionalities because the resulting clusters must have dimensionalities near l. Furthermore, like most k-means based approaches ORCLUS is in general very sensitive to noise and favors clusters of spherical shape.

4C (Computing Correlation Connected Clusters) [BKKZ04] integrates PCA into DBSCAN [EKSX96] and searches for arbitrary linear correlations of fixed dimensionality. The algorithm assigns to each point p a similarity matrix which is determined from the covariances of the vectors in the neighborhood of p. The point p is said to be a correlation core point, if the points in its neighborhood form a λ-dimensional hyperplane where λ is a user-defined parameter. The neighboring points may also deviate from the ideal plane up to a certain degree δ which is also a user defined parameter. During the run of 4C all objects are either assigned to a certain cluster or marked as noise. For each object which is not yet classified, 4C checks whether this object is a correlation core object. If the object is a correlation core object the algorithm expands the cluster belonging to this object, otherwise the object is marked as noise. To find a new cluster, 4C starts with an arbitrary correlation core object p and adds all points that are correlation-reachable from p to the current cluster. Then the algorithm continues with a point which has not yet been processed trying to expand a new cluster. Beside the dimensionality λ of the computed correlation clusters, the user must specify the additional parameters ε and μ to define the minimum density of a cluster, and the parameter δ which determines the jitter of a correlation cluster. In fact, the dimensionality of the correlation clusters produced by 4C are strongly biased towards λ. Thus, 4C has problems with correlation clusters of significantly different dimensionality. In addition, the global density threshold for correlation clusters specified by the input parameters is often hard to determine since correlation clusters of different dimensionality will most likely exhibit different densities.

CURLER (CURve cLustERs detection) [TXO05] is an algorithm for detecting arbitrary, non-linear correlations. CURLER uses the concept of micro-clusters that are generated using an EM variant and then are merged to uncover correlation clusters. CURLER improves over ORCLUS and 4C as the correlations underlying the clusters are not necessarily linear. Furthermore, as a fuzzy approach, CURLER assumes each data object to belong to all clusters simultaneously, but with different probabilities for each cluster assigned. By merging several clusters according to their co-sharing level the algorithm on the one hand becomes less sensitive to the predefined number k of clusters, thus also overcoming a severe limitation of any k-means related approach. On the other hand, the user becomes disabled to directly derive a model describing the correlations, since the original k models are no longer persistent in the resulting clustering.

DIC (Dimension Induced Clustering) [GHPT05] extends the definition of fractal correlation dimension, which measures average volume growth rate, in order to estimate the intrinsic dimensionality of the data in local neighborhoods. DIC creates a so-called local-growth model for each point. This growth model depends only on pairwise point distances and captures how each point “views” its local neighborhood. Each data object x_i is represented by a tuple (d_i, c_i), where d_i is the local dimensionality of object x_i, and c_i is the local density. Intuitively, d_i depends on the growth rate of the number of points in the neighborhood of x_i, while c_i depends on the density of points in the neighborhood of x_i. Afterwards, the EM algorithm is applied to a data set containing these tuples of each data object. As a consequence, DIC suffers from the problem that the number of clusters must be estimated in advance and specified as input parameter. In addition, DIC does not distinguish between usual (full-dimensional) clusters and correlation clusters that form a hyperplane in the data space.

Chapter 10

HiCO: Mining Hierarchies of Correlation Clusters

The detection of correlations between different features in high-dimensional data sets is a very important data mining task. These correlations can be arbitrarily complex: One or more features might be correlated with several other features, and both noise features as well as the actual dependencies may be different for different clusters. Therefore, each correlation cluster contains points that are located on a common hyperplane of arbitrary dimensionality in the data space and thus generates a separate, arbitrarily oriented subspace of the original data space. The recently proposed algorithms designed to uncover these correlation clusters have several disadvantages. In particular, these methods cannot detect correlation clusters of different dimensionality which are nested into each other. The complete hierarchical structure of correlation clusters of varying dimensionality can only be detected by a hierarchical clustering approach. Therefore, the algorithm HiCO (Hierarchical Correlation Ordering) is proposed in this Chapter, the first hierarchical approach to correlation clustering. The algorithm determines the correlation cluster hierarchy, and visualizes it using so-called correlation reachability diagrams.

The remainder of this Chapter is organized as follows: Section 10.1 introduces the basic definitions necessary for the main concepts of HiCO. To determine how to points are correlated, a special distance measurement, called correlation distance, is introduced in Section 10.2. Section 10.3 provides the new algorithm HiCO in detail. The runtime complexity of HiCO is examined in Section 10.4, whereas Section 10.5 contains an extensive experimental evaluation of HiCO. The concepts described in this chapter have been published in [ABKZ06].

10.1 Basic Definitions

In the following, $\mathcal{D}$ is assumed to be a data set of n feature vectors in a d-dimensional data space, i.e., $\mathcal{D} \subseteq \mathbb{R}^d$. The set of points $\mathcal{N}_p$ denotes the *local neighborhood* of a point $p \in \mathcal{D}$. As $\mathcal{N}_p$ is used to determine the correlation cluster to which p belongs to, $\mathcal{N}_p$ should well reflect the correlation in the local neighborhood of p. For instance, as local neighborhood of a point p the ε-neighborhood of p could be chosen. Since usually the number of objects in the ε-neighborhood of different points varies, and the query radius ε is hard to chose, it is proposed to base the local neighborhood $\mathcal{N}_p$ on the k-nearest neighbors of point p. This way, it can be ensured to consider a set of points large enough to reflect the local correlation affinity of point p well. Obviously, k should considerably exceed d. On the other hand, k should not be too large, as otherwise too many noise points may influence the derivation of the local correlation structure. Please note, that any other set of points can be chosen as local neighborhood of p, as long as it reflects the correlation in the local neighborhood of p.

Definition 10.1 (local covariance matrix).
The local covariance matrix $\mathbf{\Sigma_p}$ *of a point* $p \in \mathcal{D}$ *is formed by the local neighborhood* $\mathcal{N}_p$ *of* p*.*

Formally[1]*: Let* $\bar{x}$ *be the centroid of* $\mathcal{N}_p$*, then*

$$\mathbf{\Sigma_p} = \frac{1}{|\mathcal{N}_p|} \cdot \sum_{x \in \mathcal{N}_p} (x - \bar{x}) \cdot (x - \bar{x})^T.$$

[1]Column vectors are used by convention. Therefore, $(x - \bar{x}) \cdot (x - \bar{x})^T$ is not the inner (scalar) product but the outer product constructing a $d \times d$ matrix.

Since the local covariance matrix $\mathbf{\Sigma_p}$ *of a point* p *is a square matrix it can be decomposed into the* Eigenvalue matrix $\mathbf{E_p}$ *of* p *and the* Eigenvector matrix $\mathbf{V_p}$ *of* p *such that*

$$\mathbf{\Sigma_p} = \mathbf{V_p} \cdot \mathbf{E_p} \cdot \mathbf{V_p^T}.$$

The Eigenvalue matrix $\mathbf{E_p}$ *is a diagonal matrix holding the* d *Eigenvalues* e_i $(i = 1, \ldots, d)$ *of* $\mathbf{\Sigma_p}$ *in decreasing order in its diagonal elements. The Eigenvector matrix* $\mathbf{V_p}$ *is an orthonormal matrix with the corresponding* d *Eigenvectors of* $\mathbf{\Sigma_p}$.

Now, the *local correlation dimensionality* of a point p can be defined as the number of dimensions of the arbitrarily oriented subspace which is spanned by the major axes of the neighbors of p, $\mathcal{N}_p$. If, for instance, the objects in the local neighborhood of a point p are located near by a common line, the local correlation dimensionality of p will be 1, if the neighbors are forming a plane, the local correlation dimensionality of p will be 2, and so on. In order to get the local correlation dimensionality of a point p, the principal components of the points in $\mathcal{N}_p$ have to be determined. This is done by decomposing the local covariance matrix $\mathbf{\Sigma_p}$ into the Eigenvalue matrix $\mathbf{E_p}$ and the Eigenvector matrix $\mathbf{V_p}$ (cf. Definition 10.1). Then, the Eigenvector associated with the largest Eigenvalue has the same direction as the first principal component, the Eigenvector associated with the second largest Eigenvalue determines the direction of the second principal component, and so on. The sum of the Eigenvalues equals the trace of the square matrix $\mathbf{\Sigma_p}$ which is the total variance of the points in $\mathcal{N}_p$. Thus, the obtained Eigenvalues are equal to the variance explained by each of the principal components in decreasing order of importance. The correlation dimensionality of a point p is now defined as the smallest number of Eigenvectors explaining a portion of at least α of the total variance of the local neighbors of p.

Definition 10.2 (local correlation dimensionality).
Let $\alpha \in]0,1[$. *Then the* local correlation dimensionality λ_p *of a point* p *is the smallest number* r *of Eigenvalues* e_i *in the* $d \times d$ *Eigenvalue matrix* $\mathbf{E_p}$

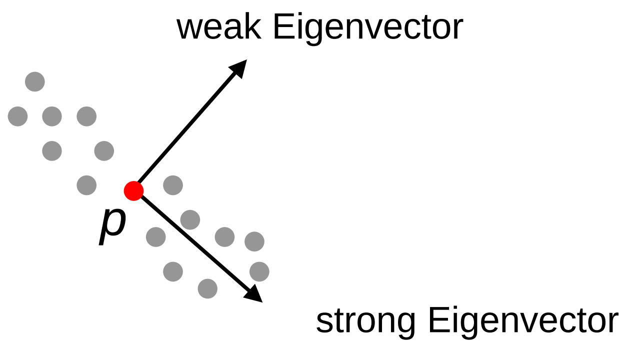


Figure 10.1: Strong and weak Eigenvectors of point p.

explaining a portion of at least α of the total variance:

$$\lambda_p = \min_{r \in \{1,\ldots,d\}} \left\{ r \,\middle|\, \frac{\sum_{i=1}^{r} e_i}{\sum_{i=1}^{d} e_i} \geq \alpha \right\}.$$

Typically α is set to values between 0.80 and 0.90. For instance, $\alpha = 0.85$ denotes that the obtained principal components of $\mathbf{\Sigma_p}$ explain 85% of the total variance. The λ_p-dimensional subspace which is spanned by the major axes of the local neighborhood of point p is denoted as the *correlation hyperplane* of p. Since it is assumed, that the Eigenvalues are ordered decreasingly in the Eigenvalue matrix $\mathbf{E_p}$, the major axes correspond to the λ_p first Eigenvectors of $\mathbf{\Sigma_p}$. The first λ_p Eigenvectors of $\mathbf{V_p}$ are also called *strong* Eigenvectors, the other Eigenvectors are called *weak* Eigenvectors. The strong Eigenvectors of point p span the hyperplane associated with a possible correlation cluster containing p, whereas the weak Eigenvectors of p are perpendicular to this hyperplane. Figure 10.1 shows a 2-dimensional example of the strong and weak Eigenvectors of a point p, where the local neighborhood of p is illustrated by the gray points around p. The strong Eigenvector of p spans the 1-dimensional line defining the correlation cluster to which p belongs, the weak Eigenvector is oriented perpendicular to the strong Eigenvector.

To easily describe some matrix computations in the following, selection matrices for strong and weak Eigenvectors are defined.

Definition 10.3 (selection matrices for strong and weak eigenvectors).
Let $p \in \mathcal{D}$ and λ_p be the local correlation dimensionality of p.

The $d \times d$ selection matrix $\check{\mathbf{E}}_\mathbf{p}$ for strong Eigenvectors *with entries $\check{e}_{ij} \in \{0,1\}$, $i,j = 1,\ldots,d$, is constructed according to the following rule:*

$$\check{e}_{ij} = \begin{cases} 1 & \textit{if} \quad i = j \leq \lambda_p \\ 0 & \textit{otherwise} \end{cases} .$$

The $d \times d$ selection matrix $\hat{\mathbf{E}}_\mathbf{p}$ for weak Eigenvectors *with entries $\hat{e}_{ij} \in \{0,1\}$, $i,j = 1,\ldots,d$, is constructed according to the following rule:*

$$\hat{e}_{ij} = \begin{cases} 1 & \textit{if} \quad i = j > \lambda_p \\ 0 & \textit{otherwise} \end{cases} .$$

The selection matrices for strong and weak Eigenvectors $\check{\mathbf{E}}_\mathbf{p}$ and $\hat{\mathbf{E}}_\mathbf{p}$ only depend on the local correlation dimensionality λ_p of point p: $\check{\mathbf{E}}_\mathbf{p}$ is a $d \times d$ diagonal matrix where the first λ_p diagonal elements are 1 and the remaining $d - \lambda_p$ diagonal element are 0 (and *vice versa* for $\hat{\mathbf{E}}_\mathbf{p}$, i.e., $\hat{\mathbf{E}}_\mathbf{p} = \mathbf{I}_{\mathbf{d}\times\mathbf{d}} - \check{\mathbf{E}}_\mathbf{p}$). Based on Definition 10.3, the *strong and weak Eigenvectors* of an object p are defined formally as follows:

Definition 10.4 (strong and weak eigenvectors).
Let $p \in \mathcal{D}$, λ_p be the local correlation dimensionality of p, and let $\mathbf{V}_\mathbf{p}$ be the corresponding Eigenvectors of point p based on the local neighborhood $\mathcal{N}_p$ of p. The first λ_p Eigenvectors of $\mathbf{V}_\mathbf{p}$ are called strong Eigenvectors*, the remaining $d - \lambda_p$ Eigenvectors are called* weak.

The matrix $\check{\mathbf{V}}_\mathbf{p}$ containing only strong Eigenvectors of p can be derived in the following way, where $\check{\mathbf{E}}_\mathbf{p}^{[1\ldots\lambda_p]}$ denotes the matrix consisting of the first λ_p columns of $\check{\mathbf{E}}_\mathbf{p}$:

$$\check{\mathbf{V}}_\mathbf{p} = \mathbf{V}_\mathbf{p} \cdot \check{\mathbf{E}}_\mathbf{p}^{[1\ldots\lambda_p]}.$$

The matrix $\hat{\mathbf{V}}_\mathbf{p}$ containing only weak Eigenvectors of p can be derived in the following way, where $\hat{\mathbf{E}}_\mathbf{p}^{[\lambda_p+1\ldots d]}$ denotes the matrix consisting of the last $d - \lambda_p$ columns of $\hat{\mathbf{E}}_\mathbf{p}$:

$$\hat{\mathbf{V}}_\mathbf{p} = \mathbf{V}_\mathbf{p} \cdot \hat{\mathbf{E}}_\mathbf{p}^{[\lambda_p+1\ldots d]}.$$

In the following, a similarity measure which reflects the distance between any vector q $\in \mathbb{R}^d$ and the correlation hyperplane exhibited by the local neighborhood of a point p is defined. Thus, this distance which is called *local correlation distance* results in 0, if vector q lies within the correlation hyperplane of p. Theoretically, the local covariance matrix $\mathbf{\Sigma_p}$ could be used as a weight matrix for the similarity measure, but it has two undesirable properties:

1. It corresponds to a similarity measure and to an ellipsoid which is oriented perpendicularly to the major axes in the local neighborhood of p. This would result in high distances to points lying within or nearby the subspace of the major axes and in low distances to points lying outside. Obviously, for detecting correlation clusters quite the opposite is needed. Figure 10.2 (left) shows the ellipsoid containing the ε-neighborhood of point p according to the local covariance matrix $\mathbf{\Sigma_p}$. Using $\mathbf{\Sigma_p}$ as weight matrix for the similarity measure, a high weight ($\frac{1}{\sqrt{\lambda_1}}$) is put on the major axis with high variance, which is defined by Eigenvector v_1. The direction with low variance, defined by Eigenvector v_2, is associated with a low weight, $\frac{1}{\sqrt{\lambda_2}}$.

2. The Eigenvalues vary with the data distribution, so some points p may have higher Eigenvalues in $\mathbf{E_p}$ than others and this would lead to incomparably weighted distances. Thus, to compute comparable local correlation distances, an inversion and a scaling of the Eigenvalues has to be performed.

Therefore, an adaption of the local covariance matrix, the so-called *local correlation similarity matrix*, is used to compute the local correlation distance between the hyperplane exhibited by the local neighborhood of p and any vector of the data space. The local similarity matrix is defined as follows:

Definition 10.5 (local correlation similarity matrix).
Let point $p \in \mathcal{D}$, $\mathbf{V_p}$ be the corresponding $d \times d$ Eigenvector matrix of the local covariance matrix $\mathbf{\Sigma_p}$ of p, and $\mathbf{\hat{E}_p}$ be the selection matrix of the weak

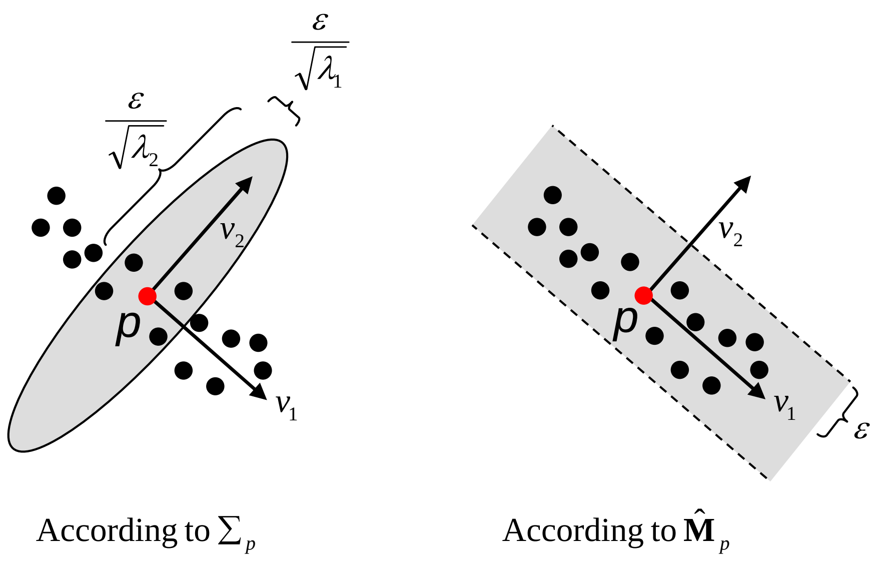

$\frac{\varepsilon}{\sqrt{\lambda_1}}$
$\frac{\varepsilon}{\sqrt{\lambda_2}}$
v_2
p
v_1
According to $\sum_p$
v_2
p
v_1
ε
According to $\hat{\mathbf{M}}_p$

Figure 10.2: ε-neighborhood of a point p (gray shaded).

Eigenvectors of p*. Then, the matrix*

$$\hat{\mathbf{M}}_\mathbf{p} = \mathbf{V}_\mathbf{p} \cdot \hat{\mathbf{E}}_\mathbf{p} \cdot \mathbf{V}_\mathbf{p}^\mathbf{T}$$

is called the local correlation similarity matrix *of* p*.*

In Figure 10.2 (right) the ε-neighborhood of a point p according to local correlation similarity matrix $\hat{\mathbf{M}}_\mathbf{p}$ of p is illustrated. Using the local correlation similarity matrix as weight matrix, the local correlation distance between the hyperplane exhibited by the neighbors of point $p \in \mathcal{D}$ and any vector $\mathrm{q} \in \mathbb{R}^d$ can be defined.

Definition 10.6 (local correlation distance).
The local correlation distance *between the hyperplane exhibited by the neighbors of point* $p \in \mathcal{D}$ *and any vector* $\mathrm{q} \in \mathbb{R}^d$ *is denoted by*

$$\textsc{LocDist}(p, \mathrm{q}) = \sqrt{\mathrm{q}^\mathrm{T} \cdot \hat{\mathbf{M}}_\mathbf{p} \cdot \mathrm{q}}.$$

The local correlation distance $\textsc{LocDist}(p, \mathrm{q})$ is the weighted Euclidean distance between vector q and the origin using the local correlation similarity matrix $\hat{\mathbf{M}}_\mathbf{p}$ as weight. Intuitively spoken, the local correlation distance

$\textsc{LocDist}(p, \mathrm{q})$ equals the Euclidean distance between vector q and the correlation hyperplane exhibited by the neighbors of p. Thus, if vector q lies within the correlation hyperplane of p, then $\textsc{LocDist}(p, \mathrm{q}) = 0$. The local correlation distance is not yet the similarity measure which is directly used in the hierarchical clustering algorithm. It is merely a construction element for the actual correlation distance measure which will be defined in the next Section.

10.2 Determination of the Correlation between two Points

In general, hierarchical clustering methods are able to find hierarchies of clusters which are nested into each other, i.e., weaker clusters in which some stronger clusters are contained. For example, the hierarchical density based clustering method OPTICS [ABKS99] is able to detect clusters of higher density which are nested in clusters of lower but still high density. The task of correlation clustering as defined in [BKKZ04] is to group those points of a data set with uniform correlation into the same correlation clusters. In the context of hierarchical correlation clustering, the hierarchy is a containment hierarchy of the correlation primitives. A hierarchy of correlation clusters means that clusters with small correlation distances, such as lines, are nested in clusters with higher correlation distances, such as planes.

The general idea is now to evaluate the correlation between two points with a special distance measure called *correlation distance*. This distance results in a small value whenever many attributes are highly correlated in the neighborhood of the two points. In contrast, the correlation distance is high if only a few attributes are highly correlated or the attributes are not correlated at all. Therefore, the strategy is to merge those points into common clusters which have small correlation distances. For instance, if two points share a common correlation line, the correlation distance between both points will be 1. If two points share a common correlation plane, they have a distance of 2, and so on. Sharing a common plane can mean different things: Both

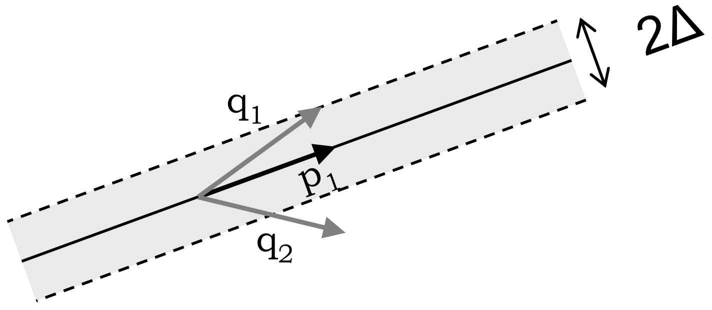


Figure 10.3: Spaces spanned by two vectors.

points can be associated to a 2-dimensional correlation plane and the planes are the same, or both points can be associated to 1-dimensional correlation lines and the lines meet at some point or are parallel but not skew.

If a distance measure with the properties mentioned above is defined for a pair of points, any of the well-known hierarchical clustering algorithms can be used to determine a hierarchy of correlation clusters. Intuitively, the distance measure between two points corresponds to the dimensionality of the data space which is spanned by the strong Eigenvectors of the two points. The notion *spanning a space* do not mean spanning in the algebraic sense of linear independence which considers two vectors to span a 2-dimensional space even if they have only a minimal difference of orientation. In this context, a vector q adds a new dimension to the space spanned by a set of vectors $\{\mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{n}\}$ if the "difference" between q and the space spanned by $\{\mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{n}\}$ is substantial, i.e., if it exceeds the threshold parameter Δ. This is illustrated in Figure 10.3: the space spanned by $\{\mathrm{q}_1\} \cup \{\mathrm{p}_1\}$ is considered to be the same as the space spanned by p_1 only. On the other hand, the set of vectors $\{\mathrm{q}_2\} \cup \{\mathrm{p}_1\}$ span a 2-dimensional space as the "difference" between q_2 and p_1 exceeds Δ.

In the following, first a definition of the *correlation dimensionality of a pair of points* $\lambda(p, q)$ is given which follows the intuition of the spanned space. Later, a method for computing this dimensionality efficiently will be proposed, given the local Eigenvector matrices and the local correlation dimensionalities of p and q. The correlation dimensionality of a pair of points

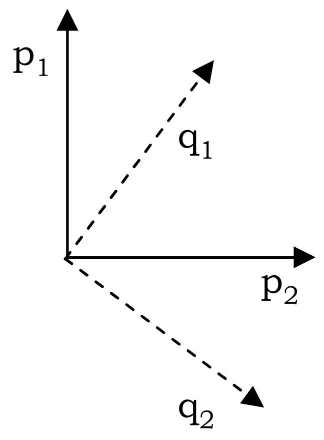


Figure 10.4: Two pairs of two vectors in a 2-dimensional vector space.

is the most important component of the correlation similarity measure and is defined as follows:

Definition 10.7 (correlation dimensionality of a pair of points). *The* correlation dimensionality *of two points $p, q \in \mathcal{D}$, denoted by $\lambda(p, q)$, is the dimensionality of the space which is spanned by the union of the strong Eigenvectors associated to p and the strong Eigenvectors associated to q.*

If the correlation dimensionality of two points p and q is determined in a strong algebraic sense, the *exactly* linearly independent vectors in the union of the strong Eigenvectors of p and q have to be computed. These n vectors form a basis of the n-dimensional subspace spanned by the strong Eigenvectors of p and q. The context of hierarchical correlation clustering does not consider the notion of the spanned space in the strong algebraic sense, but in a more informal way. That means that those vectors have to be determined that are linearly independent in the relaxed notion as mentioned above in order to allow a certain degree of jitter of data points around a perfect hyperplane.

An obvious idea for computing the correlation dimensionality of a pair of objects is to compare the *strong* Eigenvectors $\mathrm{p_i} \in \mathbf{V_p} \cdot \mathbf{\check{E}_p}$ and $\mathrm{q_i} \in \mathbf{V_q} \cdot \mathbf{\check{E}_q}$ in a one-by-one fashion. However, as Figure 10.4 shows, the two vector pairs $\{\mathrm{p_1}, \mathrm{p_2}\}$ and $\{\mathrm{q_1}, \mathrm{q_2}\}$ are linearly dependent although each vector is

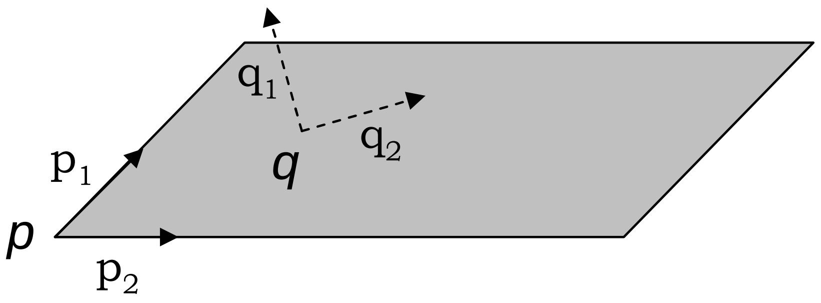


Figure 10.5: Two points with their Eigenvectors.

independent from each of the other three vectors.

One of the strong Eigenvectors, say $\mathrm{q}_1 \in \mathbf{V_q} \cdot \check{\mathbf{E}}_\mathbf{q}$, can be tested whether or not it is linearly independent - in the relaxed sense - to *all* the strong Eigenvectors $\mathrm{p_i} \in \mathbf{V_p} \cdot \check{\mathbf{E}}_\mathbf{p}$ by testing if q_1 lies (approximately) within the correlation hyperplane of p. This can be done by using the local correlation distance between the hyperplane exhibited by the neighbors of point p and vector q_1 as defined in Definition 10.6, i.e., by testing:

$$\textsc{LocDist}^2(p, \mathrm{q}_1) = {\mathrm{q}_1}^\mathrm{T} \cdot \hat{\mathbf{M}}_\mathbf{p} \cdot \mathrm{q}_1 > \Delta^2.$$

If this comparison holds, then it is known that q_1 opens up a new dimension compared to p, and that the correlation dimensionality of p and q, $\lambda(p, q)$, is at least $(\lambda_p + 1)$. But if a second vector $\mathrm{q}_2 \in \mathbf{V_q} \cdot \check{\mathbf{E}}_\mathbf{q}$ is tested, there is still the problem that q_2 can be linearly dependent from $\mathbf{V_p} \cdot \check{\mathbf{E}}_\mathbf{p} \bigcup \{\mathrm{q}_1\}$ without being linearly dependent from any vector in $\mathbf{V_p} \cdot \check{\mathbf{E}}_\mathbf{p}$ and $\{\mathrm{q}_1\}$ alone. This problem is visualized in Figure 10.5. The strong Eigenvectors q_1 and q_2 of q (depicted in dashed lines) are each linearly independent from the strong Eigenvectors p_1 and p_2 of p (depicted in solid lines) and by definition also linearly independent from each other (they are even orthogonal). However, q_2 is linearly dependent from the vectors in $\mathbf{V_p} \cdot \check{\mathbf{E}}_\mathbf{p} \bigcup \{\mathrm{q}_1\}$.

Therefore, before testing q_2, vector q_1 has to be integrated temporarily into the Eigenvector matrix $\mathbf{V_p}$, but only if q_1 indeed opens up a new dimension. To do so, the *weak* Eigenvector $\mathrm{p}_{\lambda_\mathrm{p}+1}$ has to be replaced temporarily by the new *strong* Eigenvector q_1 and afterwards the resulting matrix has to be orthonormalized.

To orthonormalize the set of vectors $\{\mathrm{p}_1, \ldots, \mathrm{p}_{\lambda_\mathrm{p}}, \mathrm{q}_1, \mathrm{p}_{\lambda_\mathrm{p}+2}, \ldots, \mathrm{p}_\mathrm{d}\}$, the following steps have to be applied according to the method of Gram-Schmitt: Note that the vector $\mathrm{p}_{\lambda_\mathrm{p}+1}$ is temporarily replaced by vector q_1, i.e., $\mathrm{p}_{\lambda_\mathrm{p}+1} = \mathrm{q}_1$.

1. $\mathrm{x}_\mathrm{i} := \mathrm{p}_\mathrm{i} - \sum_{k=1}^{i-1} \langle \mathrm{p}_\mathrm{k}, \mathrm{p}_\mathrm{i} \rangle \mathrm{p}_\mathrm{k}$ for $i = \lambda_p + 1, \ldots, d$
2. $\mathrm{p}_\mathrm{i} := \frac{\mathrm{x}_\mathrm{i}}{||\mathrm{x}_\mathrm{i}||}$ for $i = \lambda_p + 1, \ldots, d$

The resulting vectors $\{\mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{d}\}$ build now again an orthonormal basis of the d-dimensional feature space.

The orthogonalization of $(d - \lambda_p)$ vectors causes some problems because

1. the linear independence, this time in the strong algebraic sense, of the remaining Eigenvectors in $\mathbf{V_p}$ has to be guaranteed, since otherwise orthonormalization could fail.
2. the effort is considerable high as the orthonormalization (which is in $O(d^2)$) has to be performed every time a new vector is integrated into $\mathbf{V_p}$.

Therefore, the test

$$\mathrm{q}_1{}^\mathrm{T} \cdot \hat{\mathbf{M}}_\mathbf{p} \cdot \mathrm{q}_1 = \mathrm{q}_1{}^\mathrm{T} \cdot (\mathbf{V_p} \cdot \hat{\mathbf{E}}_\mathbf{p} \cdot \mathbf{V_p^T}) \cdot \mathrm{q}_1 > \Delta^2$$

will be computed in an indirect way by replacing $\hat{\mathbf{E}}_\mathbf{p}$ by $\check{\mathbf{E}}_\mathbf{p}$ which will yield the advantage that less vectors (one instead of up to d vectors) have to be orthonormalized. The justification for the indirect computation is given by the following lemma:

Lemma 10.1 (indirect similarity computation).
Let $\mathbf{V}$ *be an orthonormal matrix consisting of the strong Eigenvectors of* $\mathbf{\Sigma_p}$*,*

some of the added and orthonormalized Eigenvectors of $\mathbf{\Sigma_q}$ *and the remaining orthonormalized weak Eigenvectors of* $\mathbf{\Sigma_p}$*. Than*

$$\mathrm{x}^{\mathrm{T}} \cdot (\mathbf{V} \cdot \mathbf{\hat{E}_p} \cdot \mathbf{V^T}) \cdot \mathrm{x} = \mathrm{x}^{\mathrm{T}} \cdot x - x^{\mathrm{T}} \cdot (\mathbf{V} \cdot \mathbf{\check{E}_p} \cdot \mathbf{V^T}) \cdot x.$$

Proof.

$$\begin{aligned} \mathrm{x}^{\mathrm{T}} \cdot (\mathbf{V} \cdot \mathbf{\hat{E}_p} \cdot \mathbf{V^T}) \cdot \mathrm{x} &= \\ \mathrm{x}^{\mathrm{T}} \cdot (\mathbf{V} \cdot (\mathbf{I} - \mathbf{\check{E}_p}) \cdot \mathbf{V^T}) \cdot \mathrm{x} &= \\ \mathrm{x}^{\mathrm{T}} \cdot (\mathbf{V} \cdot \mathbf{I} \cdot \mathbf{V^T}) \cdot \mathrm{x} - \mathrm{x}^{\mathrm{T}} \cdot (\mathbf{V} \cdot \mathbf{\check{E}_p} \cdot \mathbf{V^T}) \cdot \mathrm{x} &= \\ \mathrm{x}^{\mathrm{T}} \cdot \mathrm{x} - \mathrm{x}^{\mathrm{T}} \cdot (\mathbf{V} \cdot \mathbf{\check{E}_p} \cdot \mathbf{V^T}) \cdot \mathrm{x} & \end{aligned}$$

□

The advantage of this computation is that now in the joint matrix $\mathbf{\check{M}_p} = \mathbf{V_p} \cdot \mathbf{\check{E}_p} \cdot \mathbf{V_p^T}$ the weak Eigenvectors $\mathrm{p_m}$ for $m > \lambda_p$ are not considered. Keeping the weak Eigenvectors orthonormal after every insertion of a new strong Eigenvector of $\mathbf{\Sigma_q}$ yields high computational costs: With direct computation, up to d vectors have to be orthonormalized after each insertion. Therefore, the overall complexity is $O(d^2)$ per insertion. Using the indirect computation, it is sufficient to orthonormalize only the inserted vector which can be done in $O(d)$ time. Note also that in this case the linear independence of the vector to be orthonormalized to the strong Eigenvectors in $\mathbf{V_p}$ is given, because this vector is even linear independent in the relaxed sense - and linear independency in weak sense implies linear independency in strict sense.

Now, the correlation distance between two points can be defined as follows:

Definition 10.8 (correlation distance).
The correlation distance *between two points* $p, q \in \mathcal{D}$*, denoted by* CorrDist(p, q)*, is a pair consisting of the correlation dimensionality of* p *and* q *and the Euclidean distance between* p *and* q*, i.e.,*

$$\textsc{CorrDist}(p, q) = (\lambda(p, q), \textsc{Dist}(p, q)).$$

$\textsc{CorrDist}(p, q) \leq \textsc{CorrDist}(r, s)$ *if one of the following conditions hold:*

1. $\lambda(p, q) < \lambda(r, s)$,
2. $\lambda(p, q) = \lambda(r, s)$ *and* $\textsc{Dist}(p, q) \leq \textsc{Dist}(r, s)$.

The algorithm for computing the correlation distance is presented in Figures 10.6 and 10.7. First, the correlation dimensionality $\lambda(p, q)$ of two points p and q is derived as follows: For each of the *strong* Eigenvectors $\mathrm{q_i}$ of q test whether $\mathrm{q_i}^{\mathrm{T}} \cdot \mathrm{q_i} - \mathrm{q_i}^{\mathrm{T}} \cdot (\mathbf{V_p} \cdot \mathbf{\check{E}_p} \cdot \mathbf{V_p^T}) \cdot \mathrm{q_i} > \Delta^2$. If so, increase λ_p by one and set vector $\mathrm{p}_{\lambda_\mathrm{p}}$ to the orthonormalized vector of $\mathrm{q_i}$. Finally, λ_p contains the correlation dimensionality of p and q w.r.t. p. In an analogue way, the correlation dimensionality of p and q w.r.t. q is derived. The overall correlation dimensionality $\lambda(p, q)$ is the maximum of both, λ_p, and λ_q. $\lambda(p, q)$ is now the major building block for the correlation distance. As $\lambda(p, q) \in \mathbb{N}$, many distances between different point pairs are identical. Therefore, there are many tie situations during clustering. These tie situations are resolved by additionally considering the Euclidean distance as a secondary criterion. This means, inside a correlation cluster (if there are no nested stronger correlation clusters), the points are clustered in a “conventional” way.

10.3 Algorithm HiCO

Using the correlation distance defined above as a distance measure, basically every hierarchical or even non-hierarchical clustering algorithm which is based on distance comparisons can pe performed, e.g., Single-Link [Sib73], DBSCAN [EKSX96] (non-hierarchical) or OPTICS [ABKS99]. Since OPTICS is hierarchical and more robust w.r.t. noise than Single- and Complete-Link, the algorithmic schema and visualization technique of OPTICS are used for HiCO.

In order to compute the correlation distance between two points, the following attributes have to be determined for each point p of the data set in a preprocessing step:

```
function correlationDistance(Object $p$, Object $q$, Real $\Delta$)
    $\mathbf{V_p}$ := Eigenvector matrix of $p$;
    $\lambda_p$ := local correlation dimensionality of $p$;
    $\mathbf{\check{E}_p}$ := selection matrix of strong Eigenvectors of $p$;
    $\mathbf{\check{V}_p} := \mathbf{V_p} \cdot \mathbf{\check{E}_p}^{[1 \ldots \lambda_p]}$ := matrix of strong Eigenvectors of $p$;

    $\mathbf{V_q}$ := Eigenvector matrix of $q$;
    $\lambda_q$ := local correlation dimensionality of $q$;
    $\mathbf{\check{E}_q}$ := selection matrix of strong Eigenvectors of $q$;
    $\mathbf{\check{V}_q} := \mathbf{V_q} \cdot \mathbf{\check{E}_q}^{[1 \ldots \lambda_q]}$ :=matrix of strong Eigenvectors of $q$;

    for each strong Eigenvector $\mathrm{q_i} \in \mathbf{\check{V}_q}$ do
        if $\mathrm{q_i}^{\mathrm{T}} \cdot \mathrm{q_i} - \mathrm{q_i}^{\mathrm{T}} \cdot (\mathbf{V_p} \cdot \mathbf{\check{E}_p} \cdot \mathbf{V_p^{\mathrm{T}}}) \cdot \mathrm{q_i} > \Delta^2$ then
            adjust($\mathbf{V_p}, \mathbf{\check{E}_p}, \mathrm{q_i}, \lambda_p$);
            $\lambda_p := \lambda_p + 1$;
        end if
    end for

    for each strong Eigenvector $\mathrm{p_i} \in \mathbf{\check{V}_p}$ do
        if $\mathrm{p_i}^{\mathrm{T}} \cdot \mathrm{p_i} - \mathrm{p_i}^{\mathrm{T}} \cdot (\mathbf{V_q} \cdot \mathbf{\check{E}_q} \cdot \mathbf{V_q^{\mathrm{T}}}) \cdot \mathrm{p_i} > \Delta^2$ then
            adjust($\mathbf{V_q}, \mathbf{\check{E}_q}, \mathrm{p_i}, \lambda_q$);
            $\lambda_q := \lambda_q + 1$;
        end if
    end for

    $\lambda(p,q) := \max(\lambda_p, \lambda_q)$
    return $(\lambda(p,q), \textsc{Dist}(p,q))$;
end.
```

Figure 10.6: The function to determine the correlation distance between two points.

```
procedure adjust(Matrix $\mathbf{V}$, Matrix $\check{\mathbf{E}}$, Vector $\mathrm{x}$, Integer $\lambda$)
    // let matrix $\mathbf{V}$ consist of the column vectors $\mathrm{v}_1, \ldots, \mathrm{v}_\mathrm{d}$
    // let diagonal matrix $\check{\mathbf{E}}$ consist of the diagonal elements $\check{e}_1, \ldots, \check{e}_d$

    $\mathrm{v}_{\lambda+1} := \mathrm{x}$
    $\check{e}_{\lambda+1} := 1$

    for $i = 1 \ldots \lambda$ do
        $\mathrm{v}_{\lambda+1} := \mathrm{v}_{\lambda+1} - \langle \mathrm{v}_\mathrm{i}, \mathrm{x} \rangle \cdot \mathrm{v}_\mathrm{i}$;
    end for

    $\mathrm{v}_{\lambda+1} := \frac{\mathrm{v}_{\lambda+1}}{||\mathrm{v}_{\lambda+1}||}$
end.
```

Figure 10.7: The procedure for orthonormalization.

1. The *Eigenvector matrix* $\mathbf{V_p}$ of p by decomposition of the local covariance matrix $\mathbf{\Sigma_p}$ which is the covariance matrix of the k-nearest neighbors of p.

2. The *local correlation dimensionality* λ_p which indicates the highly correlated dimensions in the k-nearest neighbors of p.

3. The *selection matrix of the strong Eigenvectors* $\check{\mathbf{E}}_\mathbf{p}$ of p which is used to compute the correlation distance between point p and other points.

As suggested in [ABKS99], a smoothing factor μ is introduced to avoid the Single-Link effect and to achieve robustness against noise points. Thus, instead of using the correlation distance $\textsc{CorrDist}(p, q)$ to measure the similarity of two points p and q, the *correlation reachability* $\textsc{CorrReach}_\mu(p, q)$ is used to compare these two points. The correlation reachability of a point q relative to a point p is defined as the maximum value of the correlation distance from p to its μ-nearest neighbor (w.r.t. the correlation distance $\textsc{CorrDist}$) and the correlation distance between p and q.

Definition 10.9 (correlation reachability).
For $\mu \in \mathbb{N}^+$, $\mu \leq |\mathcal{D}|$ let r be the μ-nearest neighbor of $p \in \mathcal{D}$ w.r.t. the

```
algorithm HiCO(Database $\mathcal{D}$, Integer $k$, Integer $\mu$, Real $\alpha$, Real $\Delta$)
    initialize empty correlation cluster order $co$;
    initialize empty priority queue $pq$ ordered by $\textsc{MinCorrReach}$;
    for each $p \in \mathcal{D}$ do
        compute $\mathbf{V_p}, \lambda_p, \check{\mathbf{E}}_\mathbf{p}$ w.r.t. parameters $k$ and $\alpha$;
        $p.\textsc{MinCorrReach} = \infty$;
        $pq$.insert($p$);
    end for
    while $pq \neq \emptyset$ do
        $p := pq$.next();
        $co$.add($p$);
        $r :=$ $\mu$-NN of $p$ w.r.t. $\textsc{CorrDist}$;
        for each $q \in pq$ do
            $cr_{new} := \max(\textsc{CorrDist}(p, r), \textsc{CorrDist}(p, q))$;
            if $cr_{new} < q.\textsc{MinCorrReach}$ then
                $q.\textsc{MinCorrReach} := cr_{new}$;
                $pq$.decrease($q$);
            end if
        end for
    end while
    return $co$;
end.
```

Figure 10.8: The HiCO algorithm.

correlation distance $\textsc{CorrDist}$. *The* correlation reachability *of a point* $q \in \mathcal{D}$ *relative to point* p *w.r.t.* μ *is defined as*

$$\textsc{CorrReach}_\mu(p, q) = \max(\textsc{CorrDist}(p, r), \textsc{CorrDist}(p, q)).$$

In each step of the algorithm, HiCO selects that point p having the minimum correlation reachability relative to any already processed point. For that purpose, each object p has an additional attribute $p.\textsc{MinCorrReach}$ that holds the minimum correlation reachability relative to any object processed before p. The pseudocode of the HiCO algorithm can be seen in Figure 10.8.

Initially, for each point $p \in \mathcal{D}$ its Eigenvector matrix $\mathbf{V_p}$, its local correlation dimensionality λ_p, its selection matrix of the strong Eigenvectors $\check{\mathbf{E}}_\mathbf{p}$ is computed. Then, a minimum correlation reachability of ∞ is assigned to each object p and p is added to a priority queue. This priority queue is organized as a heap and stores all points according to MinCorrReach in ascending order. By always taking as next point to be processed the first object in the priority queue, HiCO computes a "walk" through the data set and assigns to each remaining point q its smallest correlation reachability relative to a point considered before q in the walk. A special order of the database according to its correlation-based clustering structure is generated, the so-called *correlation cluster order* which can be displayed in a correlation reachability diagram. Such a correlation reachability diagram consists of the correlation reachability values on the y-axis of all points, plotted in the order which HiCO produces on the x-axis. The result is a visualization of the correlation clustering structure of the data set which is very easy to understand. The "valleys" in the plot represent the correlation clusters, since points within a correlation cluster typically have lower correlation reachabilities than points outside a correlation cluster.

10.4 Runtime Complexity

Let n be the number of data points and d be the dimensionality of the data space. In the preprocessing step, the Eigenvector matrix, the correlation dimensionality, and the selection matrix of the weak Eigenvectors is precomputed for each object. This step requires the determination of the local covariance matrix of an object which needs $O(n \cdot d + k \cdot d^2)$ time. Since this is done for each object in the data set the runtime complexity results in $O(n^2 \cdot d + n \cdot k \cdot d^2)$ for the determination of the local covariance matrix.

Additionally, the local covariance matrix of each point has to be decomposed into the Eigenvalue matrix and the Eigenvector matrix. This step has a complexity of $O(n \cdot d^3)$, which is absorbed by the time complexity for the determination of the local covariance matrix (since $k > d$). Overall, the complexity of the preprocessing step yields in $O(n^2 \cdot d + n \cdot k \cdot d^2)$.

During the run of HiCO, for each point p of the data set, the correlation reachability of all remaining points q in the priority queue to p has to be evaluated. This requires first the determination of the μ-nearest neighbor of p w.r.t. CorrDist which needs $O(n \cdot d^3)$ time for one object and $O(n^2 \cdot d^3)$ time for n objects. Then, the correlation reachability of q to p can be calculated in $O(d^3)$ time. Since this is done for all point pairs p and q, the complexity of the main loop of HiCO yields $O(n^2 \cdot d^3)$.

Overall, since $k < n$, the complete runtime complexity of HiCO results in $O(n^2 \cdot d^3)$.

10.5 Experimental Evaluation

All experiments have been performed on a workstation with a $2 \cdot 64$-bit 2.6 GHz CPU and 16 GB main memory. All evaluated methods have been implemented in Java. In all experiments, the input parameters of all methods have been optimized in terms of quality and the best results have been reported in order to achieve a fair comparison.

10.5.1 Results on Synthetic Data

HiCO has been applied to several synthetic data sets. In the following, the results on a 3-dimensional data set named "DS1", which is depicted in Figure 10.9, are shown. Data set "DS1" contains a hierarchy of correlation clusters consisting of two 1-dimensional correlation clusters (lines) belonging to a 2-dimensional correlation cluster (plane) and additional noise points. The correlation reachability diagram computed by HiCO with a parameter

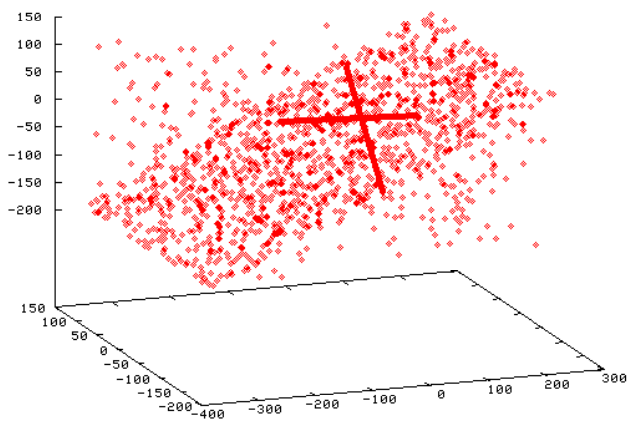
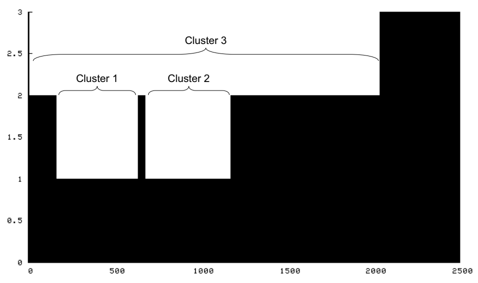


Figure 10.9: Data set "DS1".

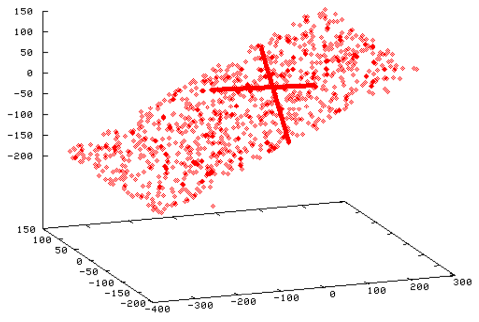
setting of $k = \mu = 20, \alpha = 0.9, \Delta = 0.05$ is shown in Figure 10.10(a). As it can be observed, HiCO detects two 1-dimensional correlation clusters that are embedded within a 2-dimensional correlation cluster. Additionally, some objects have a correlation distance of 3 which equals the data dimensionality, i.e., they can be regarded as noise. The "valleys" in the correlation reachability diagram marked with "Cluster 1", "Cluster 2", and "Cluster 3" have been analyzed. The points that are clustered together in that correlation clusters are depicted in Figures 10.10(c), 10.10(d), and 10.10(b). As it can be seen, the correlation plane "Cluster 3" corresponds to the 2-dimensional correlation cluster in the diagram, whereas the two correlation lines "Cluster 1" and "Cluster 2" exactly correspond to the 1-dimensional correlation subclusters of "Cluster 3" in the diagram. Obviously, HiCO detects the hidden correlation hierarchy exactly.

For comparison, ORCLUS, OPTICS and 4C have been applied on the same data set, but none of them were able to find the correlation clusters equally well despite reasonable parameter settings. For ORCLUS, the parameters have been set to $k = 3$ and $l = 2$, but as it can be seen in Figure 10.11, ORCLUS was not able to find the correlation clusters hidden in the synthetic 3-dimensional data set "DS1".

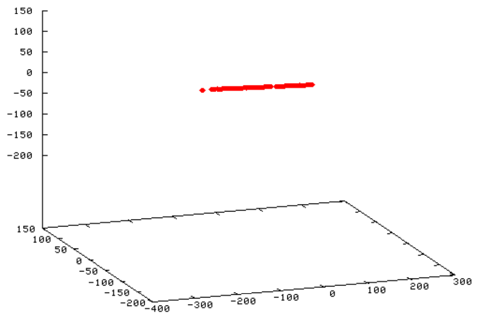
OPTICS has also been applied to the synthetic 3-dimensional data set "DS1" with a parameter setting of $\varepsilon = 1, \mu = 20$. The results are shown in



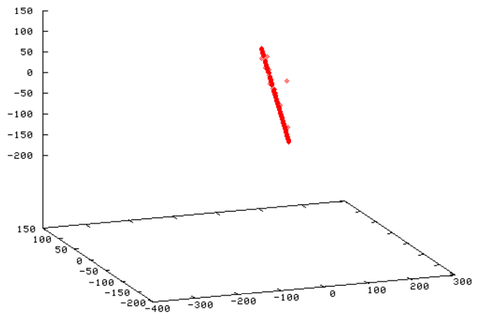
(a) Correlation reachability diagram.



(b) Cluster 3.



(c) Cluster 1.

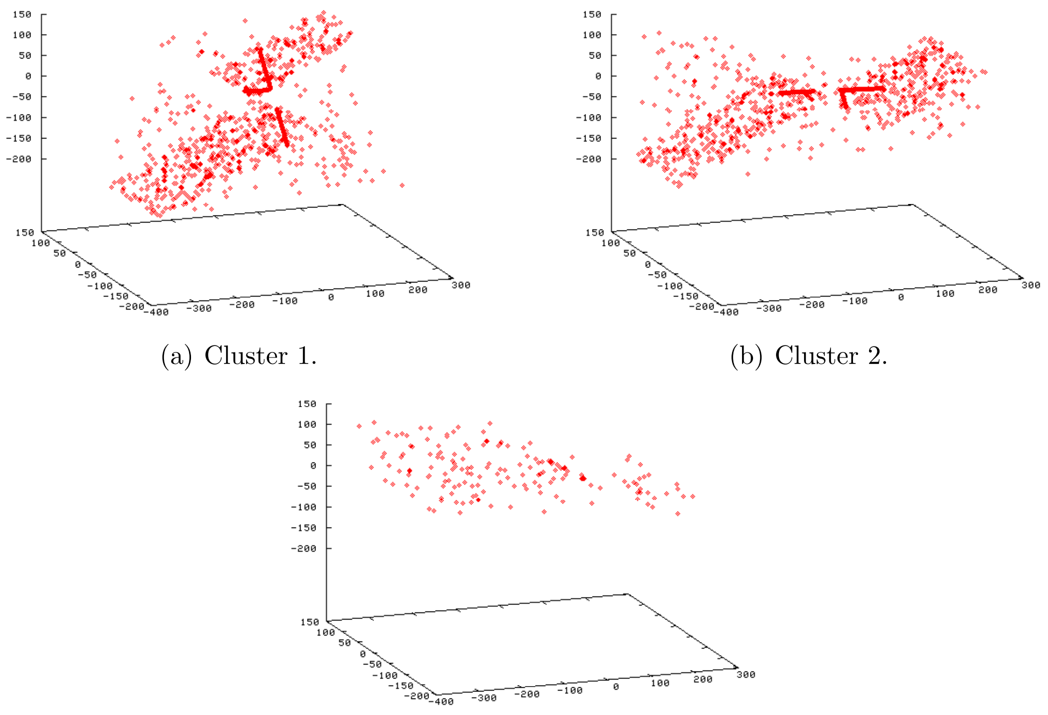


(d) Cluster 2.

Figure 10.10: Results of HiCO on “DS1”.

Figure 10.12. The reachability diagram computed by OPTICS is depicted in the upper Figure 10.12(a). The objects corresponding to the “valleys” in the reachability diagram marked with “Cluster 1” and “Cluster 2” are shown in the upper two Figures 10.12(b) and 10.12(c). As it can bee seen, OPTICS detected a hierarchy of clusters according to its density based paradigm, but it was obviously not able to separate the correlation within these clusters.

Figure 10.13 shows the results of two 4C runs with different parameter settings for parameter λ which determines the correlation dimensionality of the correlation clusters to be found. As parameter λ was set to one in the first run, 4C detected four 1-dimensional clusters consisting of the two lines in the synthetic 3D data set (cf. Figure 10.13(a)), but 4C failed to detect the 2-dimensional correlation cluster. The remaining parameters in this run were set to $\varepsilon = 0.1, \mu = 20, \delta = 0.2$. According to the parameter setting of $\lambda = 2, \varepsilon = 0.25, \mu = 20, \delta = 0.1$ in the second run, 4C found one 2-



(a) Cluster 1. (b) Cluster 2.

(c) Cluster 3.

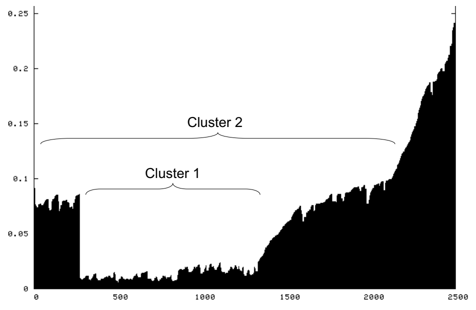
Figure 10.11: Results of ORCLUS on "DS1".

dimensional correlation cluster. This cluster consists of the plane, but also of the two lines (cf. Figure 10.13(b)). In both runs, 4C was not able to detect all three correlation clusters as HiCO did.

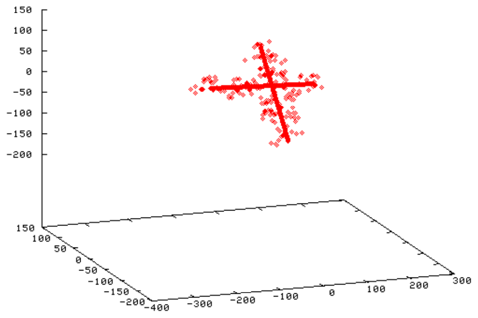
10.5.2 Results on Real-world Data

Additionally to the synthetic data sets, the effectivity of HiCO has been evaluated by using four real-world data sets. The results of HiCO applied to the real-world data sets are shown in Figure 10.14 where the correlation reachability diagrams are depicted.

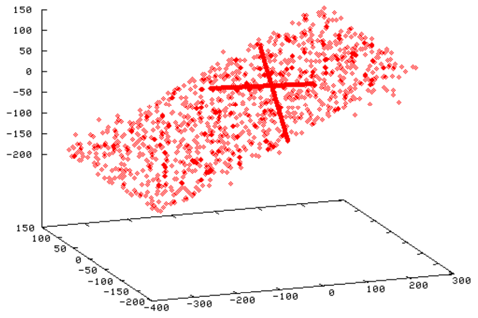
The first data set is derived from a medical study to develop screening methods in order to identify carriers of a rare genetic disorder. Four measurements were made on blood samples of approximately 150 people who do not



(a) Reachability diagram.



(b) Cluster 1.



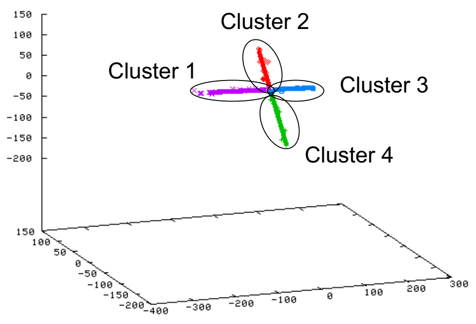
(c) Cluster 2.

Figure 10.12: Results of OPTICS on "DS1".

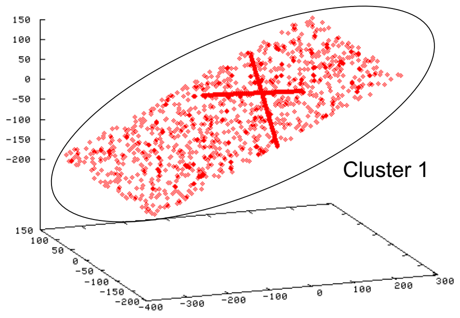
suffer from the disease and on 50 carriers of the disease. This data set is in the following called "Biomed". Applied to the "Biomed" data with a parameter setting of $k = 25, \mu = 10, \Delta = 0.25, \alpha = 0.8$, HiCO found a cluster with correlation dimensionality of 2, embedded in a larger 3-dimensional correlation cluster. The corresponding correlation reachability diagram is depicted in Figure 10.14(a). The cluster with a correlation dimensionality of 2 mostly consists of carriers of the genetic disorder. Most of the people not suffering from the disease belong to the cluster with a correlation dimensionality of 3.

As a second data set, the "El Nino" data has been used, a benchmark data set from the UCI KDD Archive[2]. The data set contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. These data are expected to aid in the understanding and prediction of El Nino / Southern Oscillation cycles. It

[2]http://kdd.ics.uci.edu/



(a) Results of 4C with $\lambda = 1$.



(b) Results of 4C with $\lambda = 2$.

Figure 10.13: Results of 4C with different λ-parameter settings on “DS1”.

contains of approximately 800 objects with 9 dimensions. The resulting correlation reachability diagram of HiCO ($k = 40, \mu = 15, \Delta = 0.25, \alpha = 0.8$) applied on the “El Nino” data set is depicted in Figure 10.14(b). As it can be seen, the hierarchy contains a 1-dimensional correlation cluster and four 2-dimensional clusters. Analyzing these clusters, it turned out that the observations belonging to these clusters were mostly made from neighbored buoys.

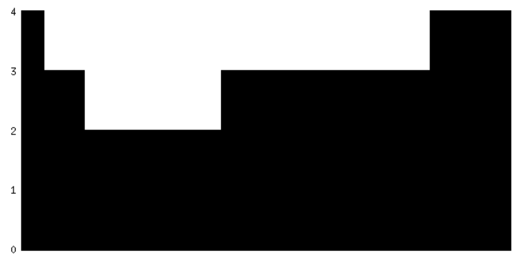
The third data set consists of 534 11-dimensional observations from the 1985 Current Population Survey[3]. It includes information for each worker: the years of education, an indicator for southern states, sex, years of working experience, an indicator for union membership, the hourly wage in dollars, age, race, occupation, sector and marital status. This data set is in the following named “Wages”. HiCO has been applied to this data set with a parameter setting of $k = 40, \mu = 10, \Delta = 0.25, \alpha = 0.8$. The resulting correlation reachability diagram is depicted in Figure 10.14(c), where a strong hierarchy of correlation clusters can be observed. HiCO computed four 2-dimensional correlation clusters embedded in a 3-dimensional correlation cluster which is also embedded in a 4-dimensional cluster. The hierarchy ends up with 5- and 6-dimensional clusters. The first of the 2-dimensional clusters consists of only white married women living not in the southern states of the USA and not belonging to any union. To the second 2-dimensional cluster male

[3]http://lib.stat.cmu.edu/datasets/CPS_85_Wages

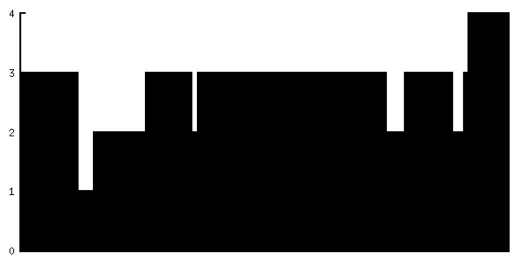
persons with the same attributes as the women in the first cluster have been assigned. The third 2-dimensional cluster consists of unmarried white women being no union member and living in the northern states. And last but not least, people belonging to the fourth 2-dimensional cluster have the same attributes as the third cluster but being men instead of women. Obviously, HiCO computed pure correlation clusters on this data set.

The fourth data set consists of the concentrations of 43 metabolites in 2,000 newborns. The newborns were labeled according to some specific metabolic diseases. Thus, the data set consists of 2,000 data points with 43 dimensions. This data set is in the following called “Metabolome”. HiCO retrieved with a parameter setting of $k = 100, \mu = 10, \Delta = 0.25, \alpha = 0.8$ on the “Metabolome” data 7- and 8-dimensional correlation clusters embedded in higher dimensional clusters. These clusters of relative low dimensionality consist of only newborns suffering from phenylketonuria (PKU), while the healthy newborns are grouped in the clusters of higher dimensionality.

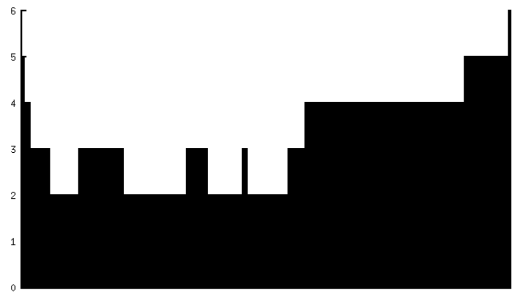
To summarize, the experiments show that HiCO detects several interesting correlation cluster hierarchies in real-world data sets.



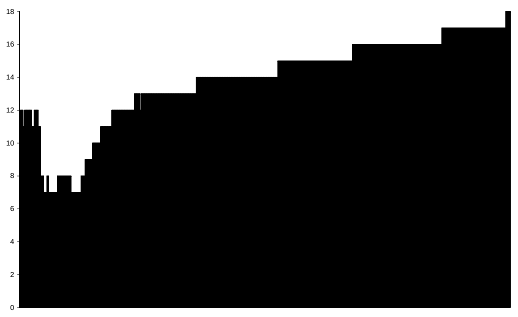
(a) Results on “Biomed” data set.



(b) Results on “El Nino” data set.



(c) Results on “Wages” data set.



(d) Results on “Metabolome” data set.

Figure 10.14: Resulting correlation reachability diagrams of HiCO applied to the real-world data sets.

Chapter 11

ERiC: Exploring Complex Hierarchical Relationships of Correlation Clusters

In high-dimensional data, clusters often only exist in arbitrarily oriented subspaces of the feature space. In addition, these so-called correlation clusters may have complex relationships between each other. For example, a correlation cluster in a 1-dimensional subspace forming a line may be enclosed within one or even several correlation clusters in 2-dimensional superspaces forming planes. In general, such relationships can be seen as a complex hierarchy that allows multiple inclusions, i.e., clusters may be embedded in several super-clusters rather than only in one. Obviously, uncovering the hierarchical relationships between the detected correlation clusters is an important information gain.

The only approach for finding hierarchies of correlation clusters introduced so far is HiCO (cf. Chapter 10). But HiCO suffers from two main drawbacks: Firstly, HiCO uses a relatively complex distance measure for every distance query in the clustering step. This results in considerable computational efforts. Secondly, the hierarchy detected by HiCO is limited to simple inclusions, i.e., HiCO cannot uncover complex hierarchies that exhibit multiple inclusions. The reason for this limitation is that HiCO uses the algorithmic schema and visualization technique of OPTICS [ABKS99]. Thus, the tree-like hierarchy of the correlation reachability plot produced by HiCO cannot reflect complex hierarchical relationships of correlation clusters like multiple inclusions. To tackle these problems, in this Chapter the algorithm ERiC (Exploring complex hierarchical Relationships among Correlation clusters) is proposed. ERiC is able to uncover efficiently complex hierarchical relationships of correlation clusters in high-dimensional data sets including not only single inclusions (like HiCO) but also multiple inclusions. In addition, ERiC provides an appropriate visualization of these complex relationships by means of a graph-based representation and outperforms HiCO and its other (non-hierarchical) competitors significantly in terms of efficiency.

The remainder of this Chapter is organized as follows. Section 11.1 describes the three steps of the new algorithm ERiC in detail: First, the partitioning of the objects w.r.t. their local correlation dimensionality. Second, the clustering of the points within each partition, and, third, the aggregation of the hierarchy of correlation clusters. The experimental evaluation in Section 11.3 shows that ERiC finds more information than state-of-the-art correlation clustering methods and outperforms existing competitors in terms of efficiency. Parts of the material presented in this Chapter have been published in [ABK+07c] and [ABK+07b].

11.1 Algorithm ERiC

Hierarchical clustering schemata such as the agglomerative schema (e.g. used by Single-Link [Sib73]), the divisive schema, or the density-based schema (e.g. used by OPTICS [ABKS99]) cannot uncover complex hierarchies that exhibit multiple inclusions. The reason for this limitation is that the resulting complex hierarchy of an algorithm implementing any of the traditional schemata is only capable of producing a tree-like hierarchy rather than producing a graph-like hierarchy. Thus, approaches like HiCO (cf. Chapter 10) that integrate a suitable "correlation distance measure" into traditional hierarchical clustering schemata cannot be used to handle hierarchies with multiple inclusions.

As a consequence, ERiC follows a different strategy. The basic idea of ERiC is first to determine all correlation clusters for all possible correlation dimensions simultaneously. Then, the hierarchical relationships among the correlation clusters is aggregated from this result. In particular, the algorithm ERiC consists of the following three steps: First, the objects of the database are partitioned w.r.t. their "local correlation dimensionality", reflecting the dimensionality of the correlation cluster in which p fits best (cf. Section 11.1.1). In a second step, the points within each partition are clustered by using a "flat" correlation clustering algorithm (cf. Section 11.1.2). The result of these two steps is the complete set of correlation clusters with the additional information regarding their correlation dimensionality. To explore the relationships among the correlation clusters found during step 2, a bottom-up strategy is applied: For any cluster $\mathcal{C}_i$ with correlation dimensionality λ_i, those clusters $\mathcal{C}_j$ with correlation dimensionality $\lambda_j > \lambda_i$ are considered as possible parents. A cluster $\mathcal{C}_j$ is a parent of $\mathcal{C}_i$ if $\mathcal{C}_i$ is embedded in (and therefore part of) $\mathcal{C}_j$. Using this information, ERiC creates the final result, i.e., a hierarchy of correlation clusters with multiple inclusions in the third step (cf. Section 11.1.3). Regarding the basic definitions applicable to this Chapter please refer to Section 10.1.

11.1.1 Partitioning w.r.t. Correlation Dimensionality

In the following, $\mathcal{D}$ is assumed to be a data set of n feature vectors in a d-dimensional data space, i.e., $\mathcal{D} \subseteq \mathbb{R}^d$. In the first step of ERiC, the objects of the data set $\mathcal{D}$ are partitioned according to their *local correlation dimensionality* as already defined in Definition 10.2. The local correlation dimensionality λ_p of a point p reflects the correlation dimensionality of the local neighborhood of p. That means, λ_p mirrors the correlation dimensionality of the correlation cluster to which p should belong to (if such a cluster exists). Thus, database objects of different local correlation dimensionality cannot form a common correlation cluster. As a consequence, it is sufficient to extract correlation clusters from each of the partitions separately.

An important aspect of Definition 10.2 is the notion of the local neighborhood of a point p, denoted by $\mathcal{N}_p$. The set of points belonging to $\mathcal{N}_p$ should reflect the correlation in the local neighborhood of p. In [BKKZ04], the correlation in the neighborhood of p is determined in terms of the ε-neighborhood of p. However, the proper representation of the local correlation is very sensitive to the choice of ε. If ε is chosen too small, $\mathcal{N}_p$ will contain an insufficient number of points, resulting in an unstable covariance matrix. As a consequence, PCA will fail to determine the proper correlation. On the other hand if ε is chosen too high, $\mathcal{N}_p$ will contain noise points that do not fit to the local correlation but are located near to p. In that case, the local correlation dimensionality of p derived by PCA of $\mathbf{\Sigma_p}$ will be considerably higher than the dimensionality of the local correlation to which p belongs. In addition, the global choice of ε as proposed in [BKKZ04] may cause that both sketched problems appear for different points in the database, i.e., for some points ε is chosen too high, whereas for some other points ε is chosen too low.

Due to these considerations, ERiC uses the k-nearest neighbors of p to determine the local correlation dimensionality of p, i.e., $\mathcal{N}_p$ contains the k-nearest neighbors of p. This ensures that the number of points in $\mathcal{N}_p$ is large enough to avoid the first problem mentioned above if k is chosen properly. Usually it seems to be a good choice to set $k = 3 \cdot d$ in order to derive a meaningful covariance matrix $\mathbf{\Sigma_p}$ and a stable singular value decomposition of $\mathbf{\Sigma_p}$ to yield its principal components. Thus, the local correlation dimensionality is well defined even for outliers. Furthermore, the range of the k-nearest neighbors is adaptive to variations of the local density: A higher local density for a point is more accurately resolved by using the k-nearest neighbors rather than the ε-neighborhood. This is due to the fact, that the ε-neighborhood would provide a considerably larger amount of points which results in in a local correlation dimensionality that does not reflect the actual local correlation dimensionality as exactly as the k-nearest neighbors do. As a consequence, the local correlation dimensionality of each point is based on an equal amount of points and, thus, it is more comparable. In fact, it empirically turned out that using the k-nearest neighbors instead of the ε-neighborhood avoids also the second mentioned problem.

After determinating the local correlation dimensionality λ_p of each object $p \in \mathcal{D}$ based on the k-nearest neighbors of p, all points $p \in \mathcal{D}$ with $\lambda_p = i$ are assigned to a partition $\mathcal{D}_i$ of the database $\mathcal{D}$. This results in a set of d disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_d$ of $\mathcal{D}$. Some of these subsets may remain empty. If not a single point exhibits a local correlation dimensionality of i, then $\mathcal{D}_i = \emptyset$. In terms of correlation clustering, $\mathcal{D}_d$ contains only noise, since there is no linear dependency of features within the neighborhood of p if $\lambda_p = d$. Note that the number of attributes involved in linear dependencies within a correlation cluster $\mathcal{C}$ is not $\lambda_{\mathcal{C}}$, but $d - \lambda_{\mathcal{C}}$.

By means of this first step of ERiC one does not only yield an appropriate local correlation dimensionality for each point in advance. Even a considerable reduction of the number of data points to be processed in each single run of a correlation clustering algorithm is given. On average, the number of data points is reduced to $\frac{n}{d}$ for each run. In fact, ERiC processes each data object only once during the second step when determining the correlation clusters.

11.1.2 Computing Correlation Clusters within each Partition

Having performed the partitioning of the database $\mathcal{D}$ in the first step, now a clustering step is performed for each partition separately. For the clustering procedure, the fact that all points within a given partition $\mathcal{D}_i$ share a common local correlation dimensionality i can be utilized. Based on the local correlation dimensionality of a point p, *strong Eigenvectors* that span the hyperplane associated with a possible correlation cluster containing p, and *weak Eigenvectors* that are perpendicular to this hyperplane are distinguished: The first λ_p Eigenvectors of the Eigenvector matrix $\mathbf{V_p}$ are called *strong Eigenvectors* of p, the remaining $d - \lambda_p$ Eigenvectors are called *weak Eigenvectors* (cf. Definition 10.4).

Two points $p, q \in \mathcal{D}_i$ are associated to the same correlation cluster if their strong Eigenvectors span approximately the same hyperplane. This will not be the case if any strong Eigenvector of p is linearly independent from the strong Eigenvectors of q or *vice versa*. However, linear dependency needs to be considered in a weakened sense to allow a certain degree of deviation, say Δ. In real-world data, it is unlikely to find a correlation cluster that perfectly fits to a hyperplane. Therefore, the strong Eigenvectors of a point q are considered to be *approximately linear dependent* from the strong Eigenvectors of p if the local correlation distance LocDist (as defined in Definition 10.6) between point p and *all* strong Eigenvectors $\mathrm{q_i}$ of point q does not exceed the threshold Δ. Formally:

Definition 11.1 (approximate linear dependency).
Let $\Delta \in]0,1[$, $p, q \in \mathcal{D}$, w.l.o.g. $\lambda_q \leq \lambda_p$, and let LocDist *denote the local correlation distance as defined in Definition 10.6. Then the strong Eigenvectors of q are* approximately linear dependent w.r.t. Δ *from the strong Eigenvectors of p if the following condition holds for* all *strong Eigenvectors $\mathrm{q_i}$ of q:*

$$\textsc{LocDist}(p, \mathrm{q_i}) \quad \leq \Delta.$$

If the strong Eigenvectors of q are approximately linear dependent w.r.t. Δ *from the strong Eigenvectors of p, it is shortly denoted by*

$$\textsc{span}(q) \subseteq^{\Delta}_{\mathrm{aff}} \textsc{span}(p).$$

Since obviously a higher dimensional subspace could never be included within a lower dimensional one, the approximate linear dependency of the strong Eigenvectors of an object q from the strong Eigenvectors of an object p is only defined for objects p, q with $\lambda_q \leq \lambda_p$. For the clustering step the condition $\lambda_q \leq \lambda_p$ of Definition 11.1 is fulfilled, because the local correlation dimensionality is the same for all objects placed in one partition $\mathcal{D}_i$. As the definition of approximate linear dependency is also needed in the third step of ERiC for aggregating the hierarchy of correlation clusters, this condition is defined more general for a different number of strong Eigenvectors λ_p and λ_q.

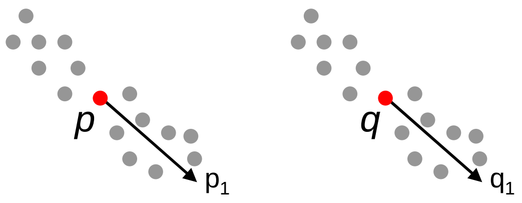


Figure 11.1: Affine subspaces.

As indicated above, the threshold Δ specifies the degree of deviation of a straight plane a correlation cluster may exhibit. Since approximate linear dependency is not symmetric in general, two objects p and q can only belong to a common correlation cluster if $\text{SPAN}(q) \subseteq^{\Delta}_{\text{aff}} \text{SPAN}(p)$ and $\text{SPAN}(p) \subseteq^{\Delta}_{\text{aff}} \text{SPAN}(q)$. However, Definition 11.1 does not take into account any affinity. Thus, the strong Eigenvectors of q are considered approximately linear dependent from the strong Eigenvectors of p, although if the space spanned by the strong Eigenvectors of q is (approximately) parallel to the space spanned by the strong Eigenvectors of p. Figure 11.1 illustrates an simple 2-dimensional example: the 1-dimensional subspace spanned by the strong Eigenvector p_1 of p is parallel to the subspace spanned by the strong Eigenvector q_1 of q, although $\text{LocDist}(p, \mathrm{q}_1) \leq \Delta$ for a small Δ and thus, $\text{SPAN}(q) \subseteq^{\Delta}_{\text{aff}} \text{SPAN}(p)$. In order to exclude affine subspaces, the distance between p and q has to be additionally assessed along the weak Eigenvectors of p, i.e., perpendicular to the hyperplane defined by the strong Eigenvectors of p. This distance which is called *affine distance* uses the local correlation similarity matrix of Definition 10.5 and is defined as follows.

Definition 11.2 (affine distance).
Let $p, q \in \mathcal{D}$, and let $\hat{\mathbf{M}}_\mathbf{p}$ be the local correlation similarity matrix of p as defined in Definition 10.5. The affine distance between p and q *is given by*

$$\text{Dist}_{\text{aff}}(p, q) = \sqrt{(p-q)^T \cdot \hat{\mathbf{M}}_\mathbf{p} \cdot (p-q)}.$$

Basically, the affine distance between a a point p and a point q is a weighted distance where the weights are based on the local neighborhood of p. The weights are constructed to take into account only the distances along the weak Eigenvectors of p, while distances along the strong Eigenvectors of

p are neglected. The affine distance as defined above is an non-symmetric distance measure, i.e., assessing the affine distance between p and q will in general not yield the same result as using the affine distance between q and p.

Combining approximate linear dependency (Definition 11.1) and the affine distance (Definition 11.2) yields the definition of a correlation distance between two points. The correlation distance between two points p and q is a binary distance measure that results in 0 if q lies within the correlation hyperplane of p and in 1, otherwise. Since both, approximate linear dependency and the affine distance are non-symmetric, the correlation distance is also a non-symmetric distance measurement.

Definition 11.3 (correlation distance).
Let $\Delta \in]0,1[$ *,* $\delta \in \mathbb{R}_0^+$*,* $p,q \in \mathcal{D}$*, and w.l.o.g.* $\lambda_q \leq \lambda_p$*. Then the* correlation distance between p and q*, denoted by* $\textsc{CorrDist}_\Delta^\delta(p,q)$*, is defined as follows*

$$\textsc{CorrDist}_\Delta^\delta(p,q) = \begin{cases} 0 & \textit{if } \text{SPAN}(q) \subseteq_{\text{aff}}^{\Delta} \text{SPAN}(p) \wedge \textsc{Dist}_{\text{aff}}(p,q) \leq \delta \\ 1 & \textit{otherwise} \end{cases} .$$

These concepts can now be integrated into a clustering algorithm which is performed on each partition. As clustering algorithm the density-based clustering algorithm GDBSCAN [SEKX98] is chosen, which is a generalization of the well-known DBSCAN clustering algorithm [EKSX96]. The choice of GDBSCAN is because of its efficiency and its effectivity. GDBSCAN is robust against noise and does not require the user to specify the number of clusters in advance.

DBSCAN iteratively performs the following procedure for each not yet processed point $p \in \mathcal{D}$: First, the ε-neighborhood of p in the feature space is computed. If this ε-neighborhood contains less than μ points, p is marked as noise and the procedure is performed for the next unclassified point in $\mathcal{D}$. Else, if p's neighborhood contains at least μ points, p is considered as *core point* and a new cluster is initiated. All points in the ε-neighborhood of p are inserted into a queue and are marked with the same cluster-ID as p. As long as this queue is not empty, the described procedure is repeated for the next point in the queue. If the queue is empty, the procedure starts with another arbitrary not yet marked point. DBSCAN terminates after a single scan over the database. $\varepsilon \in \mathbb{R}^+$ and $\mu \in \mathbb{N}^+$ are the input parameters specifying the density threshold points within a cluster must exceed.

The GDBSCAN framework as proposed in [SEKX98] provides a very easy possibility to integrate any similarity model into the algorithmic schema of DBSCAN. The basic idea is that instead of the ε-neighborhood one has to specify a generalized neighborhood of an object p, denoted by $\mathcal{N}_{NPred}(p)$, given by

$$\mathcal{N}_{NPred}(p) = \{q \mid NPred(p, q)\},$$

where $NPred(p, q)$ is a predicate on p and q that has to be reflexive and symmetric. In addition, to decide whether or not object p is a core point, a generalized minimum weight of $\mathcal{N}_{NPred}(p)$ must be defined, denoted by $MinWeight(\mathcal{N}_{NPred}(p))$.

Thus, in order to integrate the correlation distance measure into the GDBSCAN algorithm

(i) a symmetric and reflexive predicate $NPred(p, q)$ on two points $p, q \in \mathcal{D}$ and

(ii) a minimum weight $MinWeight$

have to be specified.

Intuitively, the neighborhood of an object p is formed by all points q for which the strong Eigenvectors of q span the same hyperplane as the strong Eigenvectors of p. Since the correlation distance between two points p and q can only take a value of 0 or 1 (cf. Definition 11.3), the neighborhood of an object p is formed by all points q having a correlation distance w.r.t. p of 0 to p. For symmetry reasons also the correlation distance between p and q w.r.t. q has to be 0. Formally:

Definition 11.4 (neighborhood predicate).
Let $\Delta \in]0, 1[$ *,* $\delta \in \mathbb{R}_0^+$*,* $p, q \in \mathcal{D}$*. The* neighborhood predicate *of* p *and* q *is*

given by:

$$NPred(p,q) \Leftrightarrow \textsc{CorrDist}^{\delta}_{\Delta}(p,q) = 0 \wedge \textsc{CorrDist}^{\delta}_{\Delta}(q,p) = 0.$$

The second issue is to define the minimum weight *MinWeight* on the neighborhood. Intuitively, if *MinWeight* of the neighborhood of a point p is true, p is considered as core point by the run of GDBSCAN. Analogously to traditional clustering, it is required that a point p finds at least μ points in its neighborhood $\mathcal{N}_{NPred}(p)$ using the correlation distance CorrDist as distance function.

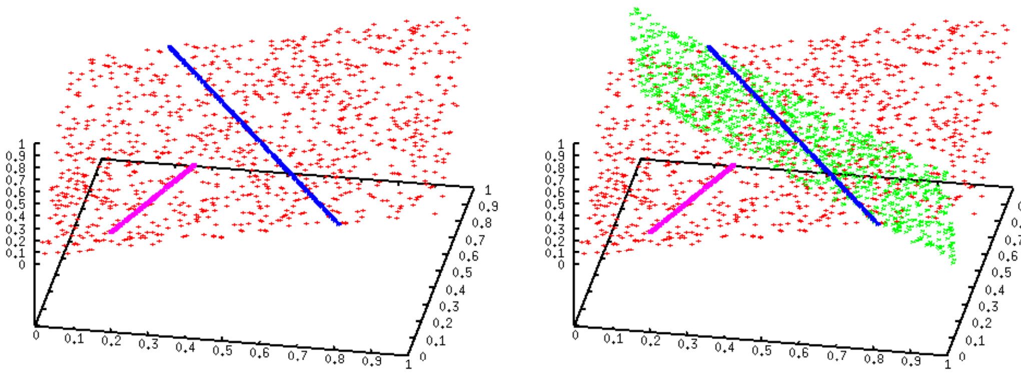
Definition 11.5 (minimum weight).
Let $\mu \in \mathbb{N}^+$, $p \in \mathcal{D}$ and let $\mathcal{N}_{NPred}(p)$ be the neighborhood of p based on the neighborhood predicate as defined in Definition 11.4. The minimum weight of $\mathcal{N}_{NPred}(p)$ such that p is a core point is given by:

$$MinWeight(\mathcal{N}_{NPred}(p)) \Leftrightarrow |\mathcal{N}_{NPred}(p)| \geq \mu.$$

Now, having defined the neighborhood predicate of an object and the minimum weight predicate of the neighborhood of an object, the GDBSCAN framework can be used to compute the set of correlation clusters in each partition $\mathcal{D}_i$.

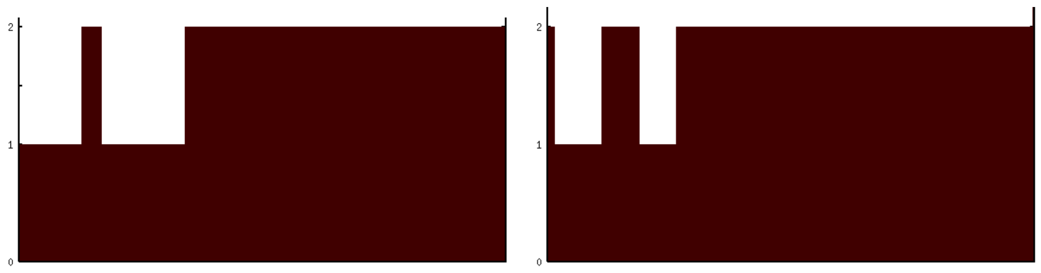
11.1.3 Aggregating the Hierarchy of Correlation Clusters

The first approach deriving information regarding the hierarchical relationships among correlation clusters is HiCO (cf. Chapter 10 for details). HiCO integrates a distance measure which takes the local correlation dimensionality of the objects into account into the hierarchical clustering algorithm OPTICS [ABKS99]. The resulting correlation reachability plot allows the user to derive a simple hierarchy of correlation clusters, but multiple inclusions cannot be derived from the resulting correlation reachability plot. Thus, the detected hierarchical structure of correlation clusters can be misleading.



(a) Data set "DSA". (b) Data set "DSB".

Figure 11.2: Data sets with different hierarchies of correlation clusters.



(a) Correlation reachability plot of "DSA". (b) Correlation reachability plot of "DSB".

Figure 11.3: Results of HiCO.

This limitation is illustrated in Figure 11.3 depicting the resulting correlation reachability plots when applying HiCO on the sample data sets from Figure 11.2. Data set "DSA" consists of a simple hierarchy of two 1-dimensional correlation clusters that are embedded within a 2-dimensional correlation cluster (cf. Figure 11.2(a)). The second data set "DSB" forms a more complex hierarchy with multiple inclusions, where one of the 1-dimensional correlation clusters is the intersection of two 2-dimensional correlation clusters, i.e., it is embedded in two clusters of higher dimensionality (cf. Figure 11.2(b)). As it can be observed, the resulting correlation reachability plots of HiCO look almost identical for both, sample data set "DSA" (cf. Figure 11.3(a)) and sample data set "DSB" (cf. Figure 11.3(b)). Since valleys in the correlation reachability plot indicate correlation clusters, both plots reveal

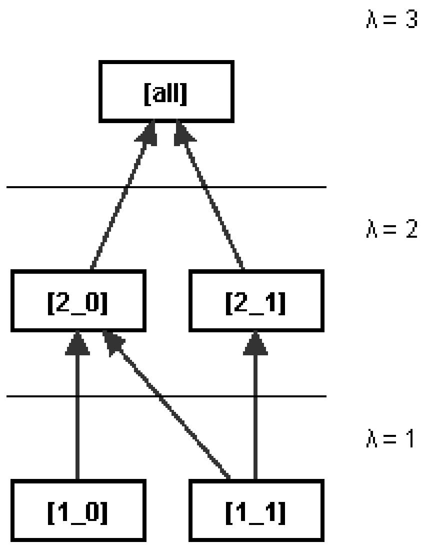


Figure 11.4: Correlation clustering graph of "DSB".

the same information of two 1-dimensional correlation clusters embedded within one 2-dimensional correlation cluster. In fact, in data set "DSB" the two 2-dimensional correlation clusters cannot be separated and the complex hierarchy consisting of the multiple inclusion cannot be detected by HiCO. The true hierarchy hidden in sample data set "DSB" can only be represented by a graph model. Figure 11.4 envisions such a visualization of the complete correlation hierarchy allowing for multiple inclusions. In fact, ERiC will produce such a visualization in the third step of the algorithm, which is described in detail in the following.

To explore the relationships among the correlation clusters found during the second step, a bottom-up strategy is now applied in the third step of the ERiC algorithm. For each correlation cluster $\mathcal{C}_i$ derived in step 2 (cf. Section 11.1.2) its centroid $\bar{c}_i$ is determined as mean value over all cluster members. Then the correlation cluster centroid $\bar{c}_i$ gets assigned its local correlation similarity matrix $\hat{\mathbf{M}}_{\bar{\mathbf{c}}_\mathbf{i}}$ (cf. Definition 10.5) using all cluster members as neighborhood $\mathcal{N}_{\bar{c}_i}$. Note that the correlation dimensionality of correlation cluster $\mathcal{C}_i$, and thus the local correlation dimensionality of centroid $\bar{c}_i$ is already given by the partition $\mathcal{D}_i$ to which $\mathcal{C}_i$ belongs to, i.e., $\lambda_{\mathcal{C}_i} = \lambda_{\bar{c}_i} = i$.

```
procedure buildHierarchy(ClusterList $cl$, Real $\Delta$, Real $\delta$)
  $\lambda_{max} := d$; // $d$ = dimensionality of data space
  for each $\mathcal{C}_i \in cl$ do
    $\bar{c_i} := \mathcal{C}_i$.centroid;
    for each $\mathcal{C}_j \in cl$ with $\lambda_{\mathcal{C}_i} < \lambda_{\mathcal{C}_j}$ do
      $\bar{c_j} := \mathcal{C}_j$.centroid;
      if $\lambda_{\mathcal{C}_j} = \lambda_{max} \wedge \mathcal{C}_i$.parents=$\emptyset$ then
        $\mathcal{C}_i$.addParent($\mathcal{C}_j$);
      else
        if $\textsc{CorrDist}^{\delta}_{\Delta}(\bar{c_j}, \bar{c_i}) = 0 \wedge$
          ($\mathcal{C}_i$.parents=$\emptyset \vee \neg$ isParent($\mathcal{C}_j$, $\mathcal{C}_i$.parents, $\Delta$, $\delta$))
        then
          $\mathcal{C}_i$.addParent($\mathcal{C}_j$);
        end if
      end if
    end for
  end for
end.
```

Figure 11.5: The procedure to build the hierarchy of correlation clusters.

As mentioned before, the parent of a cluster $\mathcal{C}_i$ with correlation dimensionality $\lambda_{\mathcal{C}_i}$ can be any cluster $\mathcal{C}_j$ with correlation dimensionality $\lambda_{\mathcal{C}_j} > \lambda_{\mathcal{C}_i}$. Therefore, the following steps are applied to each correlation cluster $\mathcal{C}_i$ and each correlation cluster $\mathcal{C}_j$ with $\lambda_{\mathcal{C}_j} > \lambda_{\mathcal{C}_i}$: If no parents have already been assigned to correlation cluster $\mathcal{C}_i$, and correlation cluster $\mathcal{C}_j$ is the noise cluster, i.e., the correlation dimensionality $\lambda_{\mathcal{C}_j}$ of $\mathcal{C}_j$ equals the dimensionality of the feature space, correlation cluster $\mathcal{C}_j$ will become the only parent of correlation cluster $\mathcal{C}_i$.

Otherwise it is checked if both correlation clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ together form a $\lambda_{\mathcal{C}_j}$-dimensional correlation cluster. This is given if the correlation

```
function isParent(Cluster $\mathcal{P}$, ClusterList $cl$, Real $\Delta$, Real $\delta$)
    for each $\mathcal{C} \in cl$ do
        $\bar{p} := \mathcal{P}$.centroid;
        $\bar{c} := \mathcal{C}$.centroid;
        if $\textsc{CorrDist}_{\Delta}^{\delta}(\bar{p}, \bar{c}) = 0$ then
            return true;
        end if
    end for
    return false;
end.
```

Figure 11.6: The function to check whether a cluster is a parent of one of the clusters in a list.

distance between the centroid of $\mathcal{C}_j$ and the centroid of $\mathcal{C}_i$ equals 0, i.e., if $\textsc{CorrDist}_{\Delta}^{\delta}(\bar{c}_j, \bar{c}_i) = 0$. Please note that now the correlation distance only has to be considered w.r.t. the centroid of the higher dimensional (potential) parent cluster $\mathcal{C}_j$, since obviously, a higher dimensional subspace could never lie within a lower dimensional one. If $\textsc{CorrDist}_{\Delta}^{\delta}(\bar{c}_j, \bar{c}_i) = 0$, correlation cluster $\mathcal{C}_j$ will become a parent of correlation cluster $\mathcal{C}_i$, if one of the following conditions holds: Either $\mathcal{C}_i$ has no parents so far or $\mathcal{C}_j$ is no parent cluster of the already assigned parents of correlation cluster $\mathcal{C}_i$, because in that case the relationship between $\mathcal{C}_i$ and $\mathcal{C}_j$ is that of a grandparent. The methods used to build the correlation hierarchy from the correlation clusters are depicted in Figure 11.5 and Figure 11.6.

The resulting hierarchy is visualized by using a graph-like representation, the so-called *correlation clustering graph*. An example is depicted in Figure 11.4, showing the hierarchy of correlation clusters in sample data set "DSB" (cf. Figure 11.2(b)). In general, the representation is organized top-down w.r.t. the correlation dimensionality similar to a tree but allows multiple inclusions. The "root" of the correlation clustering graph contains all objects in partition $\mathcal{D}_d$. All correlation clusters with equal correlation dimensionality

```
algorithm ERiC(Database $\mathcal{D}$, Integer $k$, Integer $\mu$, Real $\alpha$, Real $\Delta$, Real $\delta$)
    // Step 1: Partition data objects according
    // to their local correlation dimensionality
    initialize $\mathcal{D}_1, \ldots, \mathcal{D}_d$ with $\mathcal{D}_i = \emptyset$;
    for each $p \in \mathcal{D}$ do
        $\mathcal{N}_p :=$ NN$_k(p)$;
        compute $\lambda_p$ w.r.t. NN$_k(p)$ and $\alpha$;
        $\mathcal{D}_{\lambda_p} := \mathcal{D}_{\lambda_p} \cup p$;
    end for

    // Step 2: Extract correlation clusters from each partition
    initialize empty list of clusters $cl$;
    for each $\mathcal{D}_i \in \mathcal{D}_1, \ldots, \mathcal{D}_d$ do
        $cl_i :=$ GDBSCAN($\mathcal{D}_i$, $NPred$, $MinWeight$);
        $cl$.add($cl_i$);
    end for

    // Step 3: Aggregate hierarchy of correlation clusters
    buildHierarchy($cl$, $\Delta$, $\delta$);
end.
```

Figure 11.7: The ERiC algorithm.

are placed at the same level below the root. Thus, 1-dimensional correlation clusters are placed at the bottom level. Each object is placed in that correlation cluster with the smallest correlation dimensionality. An edge between two nodes of the correlation clustering graph indicates a containment relationship. In fact, a node N represents a correlation cluster of all objects assigned to N as well as all objects assigned to child nodes of N.

The overall procedure of ERiC is visualized in Figure 11.7: Step 1 partitions the database points according to their local correlation dimensionality. The k-nearest neighbors of p are chosen as neighborhood $\mathcal{N}_p$ of a point p. Step 2 applies GDBSCAN with *NPred* and *MinWeight* as defined in Definitions 11.4 and 11.5, respectively. Then, the third step builds the hierarchy as described above.

11.2 Runtime Complexity

Runtime Complexity. Let n be the number of data points and d be the dimensionality of the data space. The preprocessing step of ERiC works for each point as follows: First, a k-nearest neighbor query based on the Euclidean distance is performed which has a complexity of $O(n \cdot d)$ since the data set is scanned sequentially. Based on the result of the k-nearest neighbor query, the $d \times d$ covariance matrix is determined. This can be done in $O(k \cdot d^2)$ time. Then the covariance matrix is decomposed by using PCA which requires $O(d^3)$ time. Thus, for all points together the time complexity results in $O(n^2 \cdot d + n \cdot k \cdot d^2)$ in the first step of ERiC, since $k > d$.

The original GDBSCAN has a worst case complexity of $O(n^2)$ on top of the sequential scan. Applying the correlation distance as given in Definition 11.3, the time complexity of the second step of ERiC is $O(n_p^2 \cdot d^3)$ for each partition of the data set, where n_p denotes the number of objects in the each particular partition. Assuming that the data points are uniformly distributed over all possible correlation dimensionalities, and all partitions will contain $\frac{n}{d}$ points, the second step of ERiC yields an average time complexity of $O(n^2 \cdot d^2)$.

The hierarchy aggregation considers all pairs of correlation clusters $(\mathcal{C}_i, \mathcal{C}_j)$ associated to different partitions with $\lambda_i < \lambda_j$, and determines the correlation distance CorrDist for the corresponding correlation cluster centroids. Thus, an upper bound for the complexity of the third step corresponds to $O(|\mathcal{C}|^2 \cdot d^3)$, where $|\mathcal{C}|$ is the number of correlation clusters. However, in the experimental evaluation it is shown that the third step requires only a marginal runtime compared to the overall runtime of the ERiC algorithm. This is due to the fact that $|\mathcal{C}| << n$ holds.

Thus, the overall time complexity of ERiC on top of the sequential scan of the data set can be considered as $O(n^2 \cdot d^2)$.

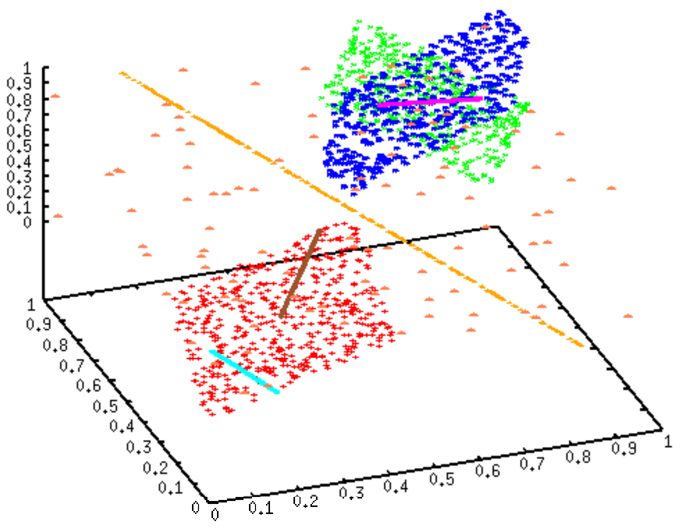


Figure 11.8: Data set “DS1”.

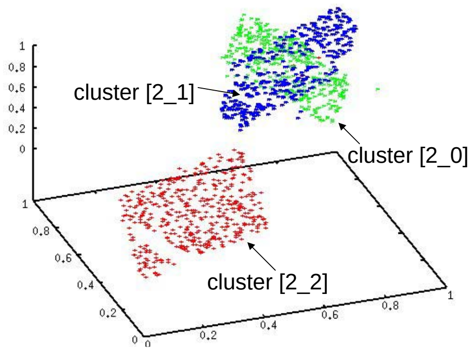
11.3 Experimental Evaluation

All experiments have been performed on a workstation with a $2 \cdot 64$-bit 2.6 GHz CPU and 16 GB main memory. All evaluated methods have been implemented in Java. In all experiments, the input parameters of all methods have been optimized in terms of quality and the best results have been reported in order to achieve a fair comparison.

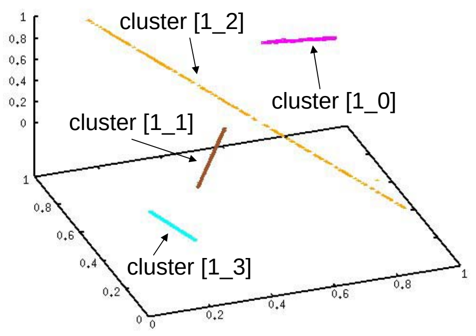
11.3.1 Effectivity

Synthetic Data Set. The accuracy of ERiC in comparison to ORCLUS, 4C, and HiCO has been evaluated on several synthetic data sets. Exemplarily, the results on one data set named “DS1” are shown. The synthetic data set contains 3-dimensional objects grouped in a complex hierarchy of arbitrarily oriented correlation clusters with multiple inclusion and noise points. The attribute values of the synthetic data set are in the range of 0.0 to 1.0.

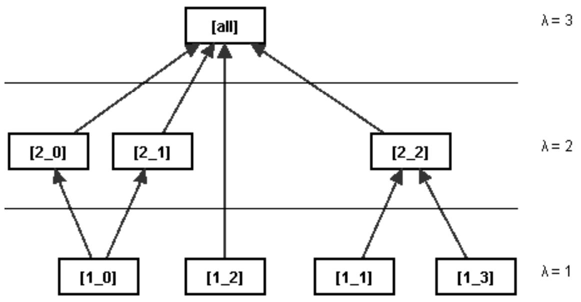
The synthetic data set “DS1” (cf. Figure 11.8) contains 3-dimensional objects grouped in a complex hierarchy of four 1-dimensional and three 2-



(a) 2-dimensional correlation clusters.



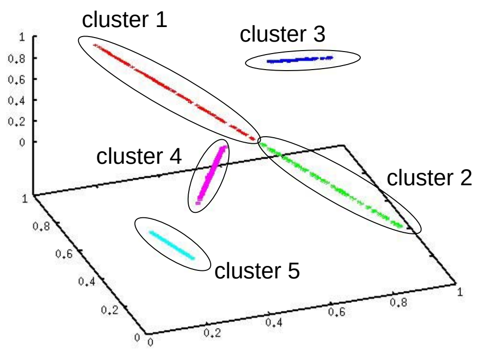
(b) 1-dimensional correlation clusters.



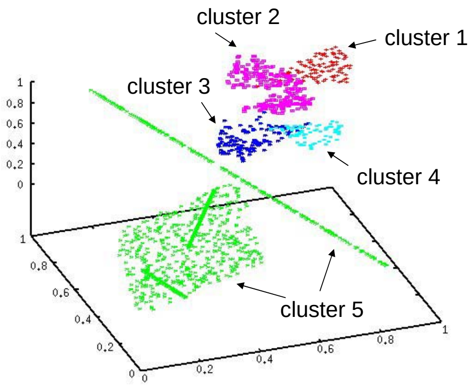
(c) Correlation clustering graph.

Figure 11.9: Results of ERiC on "DS1".

dimensional correlation clusters with a multiple inclusion and some noise points. The results of ERiC applied to "DS1" using a parameter setting of $k = 16, \mu = 30, \alpha = 0.85.\Delta = 0.1$ are shown in Figure 11.9. In the upper left Figure 11.9(a) the three 2-dimensional correlation clusters found by ERiC are marked with different colors, the upper right Figure 11.9(b) shows the four 1-dimensional correlation clusters found by ERiC. In the lower Figure 11.9(c) the resulting hierarchy visualized by the correlation clustering graph is depicted. As it can be seen, the graph illustrates the correct and complete hierarchy. One can see at a glance that the data set contains two 1-dimensional clusters (lines "1_1" and "1_3") embedded within a 2-dimensional cluster (plane "2_2"), one separate 1-dimensional cluster (line "1_2"), and a multiple inclusion of one 1-dimensional cluster (line "1_0") embedded within two 2-dimensional clusters (planes "2_1" and "2_2").



(a) Results of 4C with $\lambda = 1$.

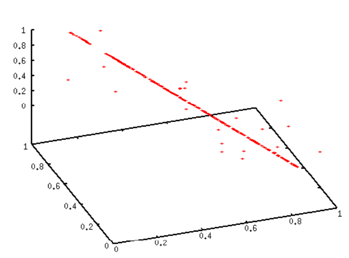


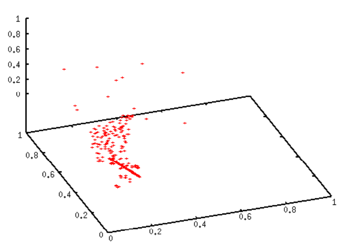
(b) Results of 4C with $\lambda = 2$.

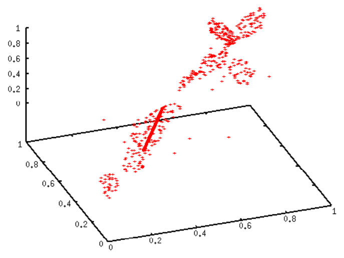
Figure 11.10: Results of 4C with different λ-parameter settings on "DS1".

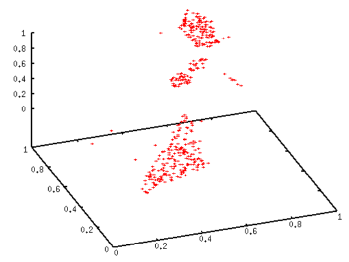
For comparison, ORCLUS, 4C, and HiCO have also been applied to data set "DS1", but none of the existing state-of-the-art correlation clustering approaches performs equally well. The algorithm 4C can produce a "flat" clustering, i.e., 4C can either detect the 1-dimensional correlation clusters or the 2-dimensional one, but not both within a single run. The results of 4C with different settings for parameter λ which is an upper bound for the correlation dimensionality of the clusters to be found, are depicted in Figure 11.10. The left Figure 11.10(a) shows the five 1-dimensional correlation clusters found by 4C with a parameter setting of $\lambda = 1, \varepsilon = 0.05, \mu = 10, \delta = 0.2$. As it can be seen, 4C splits the compact cluster "1_2" (shown in Figure 11.9) into two clusters. The three 2-dimensional planes have been classified as noise in this run. In the right Figure 11.10(b) the five 2-dimensional correlation clusters detected by 4C with a parameter setting of $\lambda = 2, \varepsilon = 0.1, \mu = 25, \delta = 0.1$ is shown. In this run, on the one hand, 4C has problems to separate the 1-dimensional correlation clusters "1_1", "1_2", and "1_3" from the 2-dimensional correlation cluster "2_2" as ERiC did (cf. Figure 11.9). On the other hand, 4C splits compact clusters into several parts, e.g., clusters "1_2", "2_0", and "1_0". When looking at the results of ORCLUS on "DS1" ($k = 7, l = 2$) which are depicted in Figure 11.11, one can see that ORCLUS completely failed to detect all correlation clusters in data set "DS1".

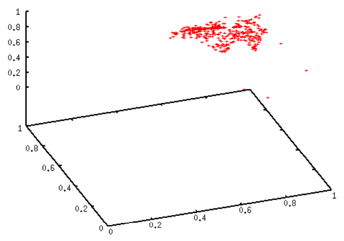
Since both 4C and ORCLUS produce a flat clustering, no hierarchy can

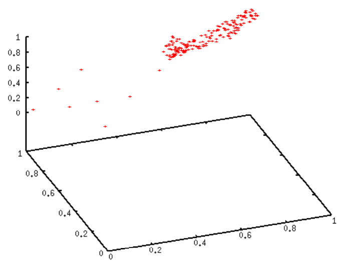

(a) ORCLUS - cluster 1.


(b) ORCLUS - cluster 2.


(c) ORCLUS - cluster 3.


(d) ORCLUS - cluster 4.


(e) ORCLUS - cluster 5.


(f) ORCLUS - cluster 6.

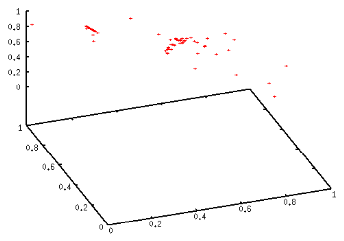

(g) ORCLUS - cluster 7.

Figure 11.11: Results of ORCLUS on "DS1".

be derived from their results. Last but not least, the result of HiCO with a parameter setting of $k = 16, \mu = 30, \alpha = 0.85, \Delta = 0.1$ on "DS1" is depicted in Figure 11.12. The obtained correlation reachability diagram has been analyzed and the objects in the "valleys" have been marked with the according cluster memberships. As it can be seen, HiCO can detect the simple hierarchical relationships, but the multiple inclusion of cluster "1_0" in cluster "2_0" cluster "2_1" is not visible at all in the resulting correlation plot. In summary, while ERiC has no problems to reveal the complete hierarchy of correlation clusters and to detect all correlation clusters correctly, the

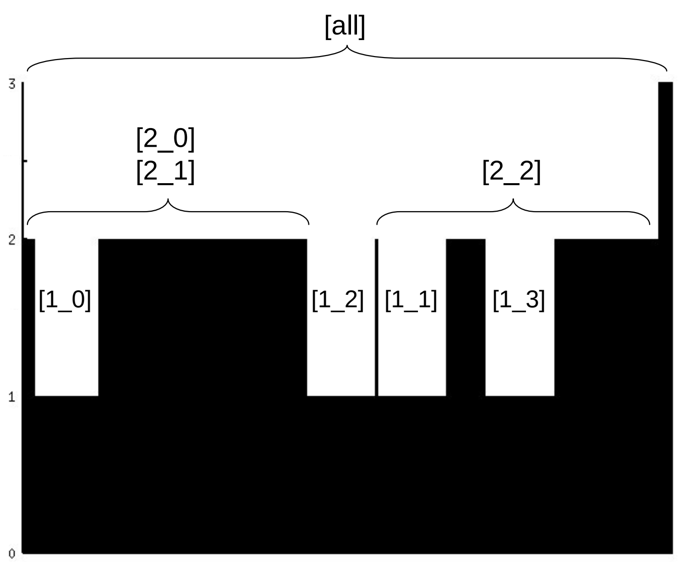


Figure 11.12: Result of HiCO on “DS1”.

competitors all fail to produce the true clusters and the proper hierarchy.

Real-world Data Sets. Additionally to the synthetic data set, the effectivity of ERiC has been evaluated by using several real-world data sets. First, ERiC has been applied on the “Wages” data set[1] consisting of 534 11-dimensional observations from the 1985 Current Population Survey. Since most of the attributes are not numeric, only 4 dimensions (YE=years of education, W=wage, A=age, and YW=years of work experience) have been used for clustering. The parameters of ERiC were chosen to $k = 5, \mu = 4, \alpha = 0.85, \Delta = 0.01$. The results are shown in Figure 11.13(a). ERiC found seven correlation clusters. The two one-dimensional correlation clusters “1_0” and “1_1” contain both the data of people having 12 years of education. The people in the first correlation cluster are all of age 22 and have a working experience of 4 years. The second 1-dimensional correlation cluster contains people at the age of 38 with a working experience of 16 years. The four 2-dimensional correlation clusters found by ERiC consist of people having constant years of education and a linear dependency between their age and

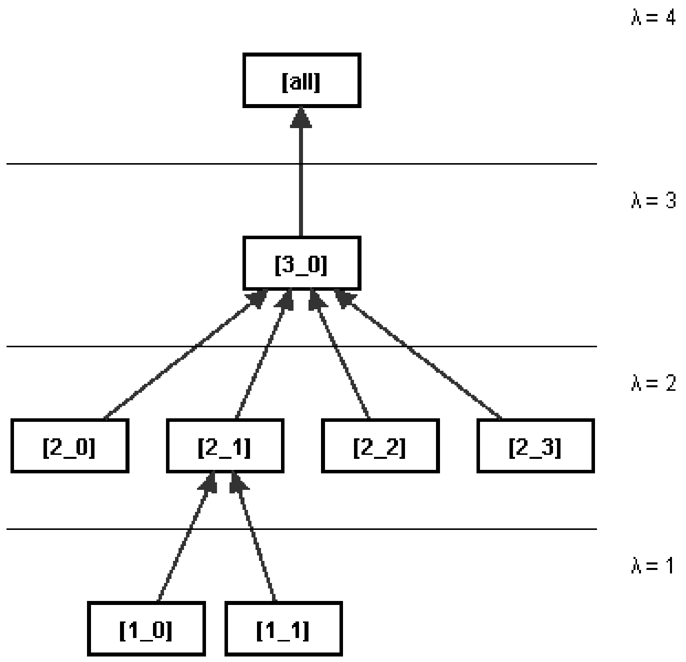
[1] http://lib.stat.cmu.edu/datasets/CPS_85_Wages

their years of working experience. In the 3-dimensional correlation cluster "3_0" those employees are grouped which started school at the age of 6 years and after graduation immediately began working. Thus, the years of education equals the difference of the age, the years of working experience and 6. The contents of the correlation clusters are summarized in Figure 11.13(b). Neither HiCO, 4C nor ORCLUS were able to detect meaningful correlation clusters in the "Wages" data set.

Then, ERiC has been applied to the (original) Wisconsin "Breast Cancer" Database from the UCI ML Archive[2]. This data set consists of 683 patients suffering from two types of breast cancer, benign and malignant. Each patient is represented by a 9-dimensional vector of specific biomedical features. ERiC detected four almost pure correlation clusters in this data set. The hierarchy generated by ERiC on this data set with a parameter setting of $k = 30, \mu = 30, \alpha = 0.85, \Delta = 0.75$ is depicted in Figure 11.14. The resulting hierarchy contains four correlation clusters that are placed in two different branches in the graph. It is worth mentioning that the two lower dimensional correlation clusters "2_0" and "3_0" in the first branch are pure clusters, i.e., they only contain benign patients. The higher dimensional correlation cluster "5_0" and its parent cluster "6_0" in the second branch are nearly pure, they contain almost only malignant patients. Some patients from both classes could not be separated and were labeled as noise. Again ERiC outperforms its competitors, since none of them were able to detect pure correlation clusters in this data.

A third real-world data set used for evaluating ERiC is the "Pendigits" data set[3] containing approximately 7,500 16-dimensional points, representing certain features of hand-written digits. The objects are labeled according to the digit. The resulting hierarchy computed by ERiC with a parameter setting of $k = 15, \mu = 10, \alpha = 0.85, \Delta = 0.5$ is depicted in Figure 11.15. Interestingly, all clusters found by ERiC are pure, i.e., contain only objects from one class. The clusters forming the observed multiple inclusion also contain objects from the same class.

[2] http://www.ics.uci.edu/~mlearn/MLSummary.html
[3] http://www.ics.uci.edu/~mlearn/databases/pendigits/



(a) Hierarchy generated by ERiC

cluster	description
1_0	YE = 12, A = 22, YW = 4
1_1	YE = 12, A = 38, YW = 20
2_0	YE = 14, A = YW + 20
2_1	YE = 12, A = YW+18
2_2	YE = 16, A = YW + 22
2_3	YE = 13, A = YW+19
3_0	YE = A - YW - 6

(b) Contents of found clusters

Figure 11.13: Results of ERiC on "Wages" data.

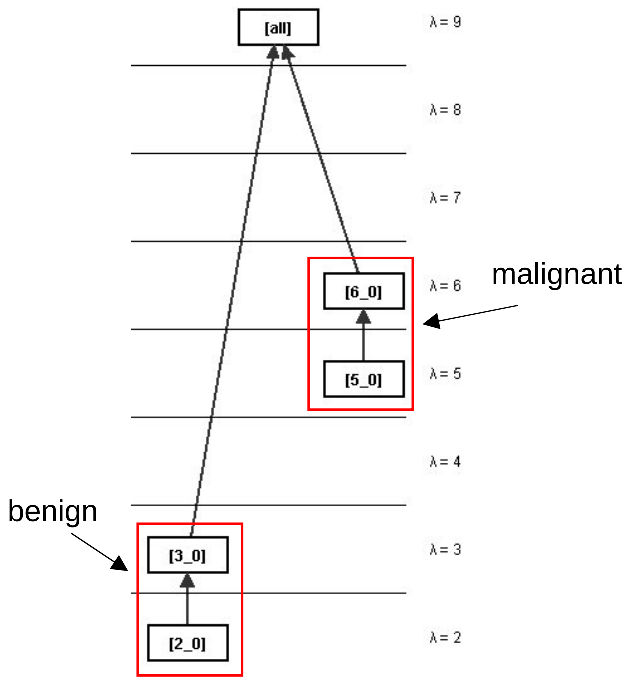


Figure 11.14: Results of ERiC on "Breast Cancer" data.

In summary, the experiments confirmed that ERiC finds meaningful cluster hierarchies allowing for multiple inclusions in real-world data sets.

11.3.2 Efficiency

For the evaluation of efficiency, synthetic data sets have been used where the dimensionality or the number of points has been varied.

For the impact of the dimensionality of the data space on the runtime, 10 data sets with a varying dimensionality of $d = 10, 20, 30, \ldots, d_{max} = 100$ have been created. For each data set, 10,000 objects were equally distributed over 10 correlation clusters, where the single attributes have values in the range of $[0.0, 1.0]$. In the first experiment, the runtime of ERiC, HiCO, 4C and ORCLUS has been compared w.r.t. the dimensionality of the data set. The parameters for ERiC were set to $k = 50, \mu = 500, \alpha = 0.999, \Delta = 0.01$. HiCO has been applied to the data sets with a parameter setting of $k = 50, \mu =$

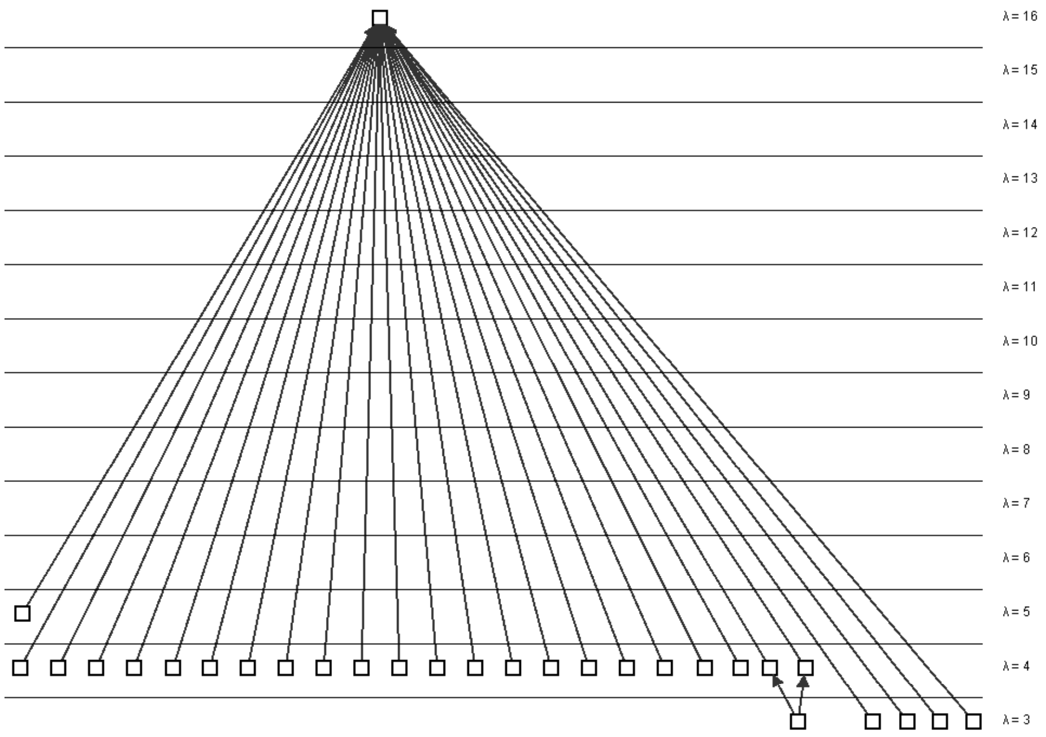


Figure 11.15: Results of ERiC on "Pendigits" data.

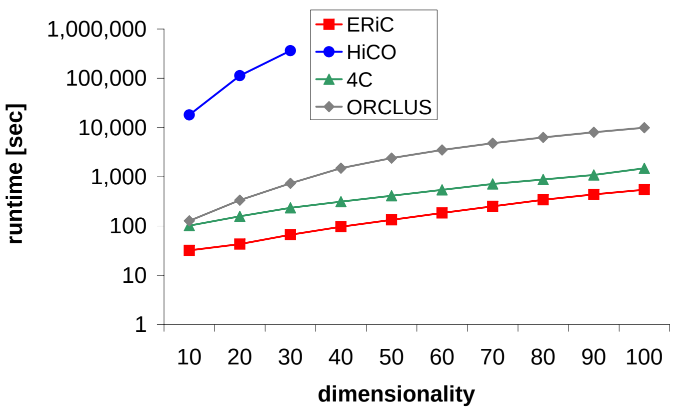


Figure 11.16: Runtime of ERiC, HiCO, 4C, and ORCLUS w.r.t. the dimensionality.

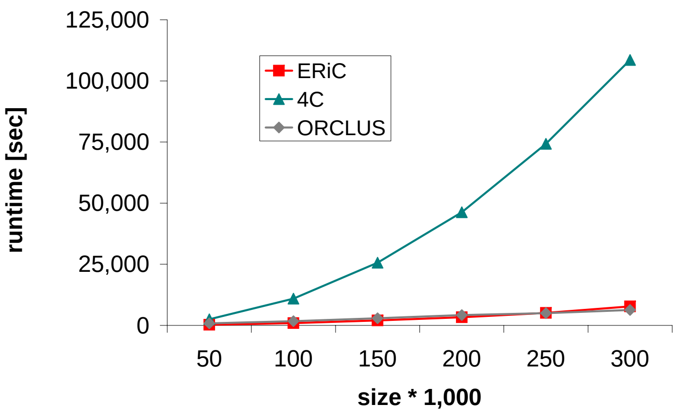


Figure 11.17: Runtime of ERiC, 4C, and ORCLUS w.r.t. the size.

$500, \alpha = 0.999, \Delta = 0.01$. The λ parameter of 4C was set to the maximal occurring correlation dimensionality, i.e., $\lambda = d - 1$. The parameter μ which determines the minimum number of objects within a correlation cluster was set to $\mu = 500$. The remaining parameters were set to $\varepsilon = 0.1, \delta = 0.01$. As a fair setting, the parameter k of ORCLUS was set to the exact number of correlation clusters in the data set and parameter l was set to the maximal occurring correlation dimensionality, i.e., $k = 9$ and $l = d - 1$. As it can be seen in Figure 11.16, ERiC clearly outperforms the other competitors (please note the logarithmic scale of the runtime-axis).

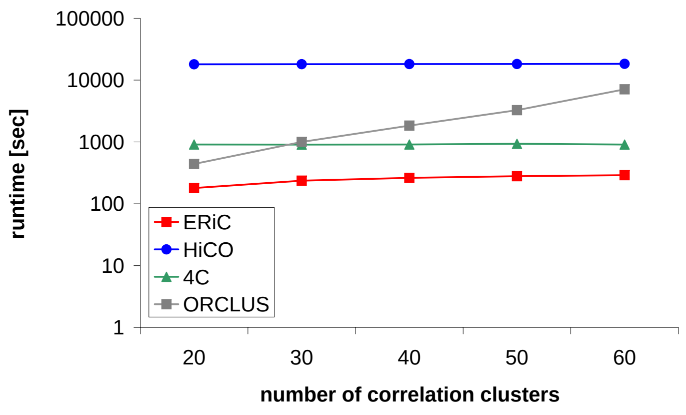


Figure 11.18: Runtime of ERiC, HiCO, 4C and ORCLUS w.r.t. the number of clusters.

Analogously, for the impact of the size of the data set on the runtime, six data sets of dimensionality $d = 10$ have been created with an increasing number of objects ranging from 50,000 to 300,000. The objects are equally distributed over nine correlation clusters of correlation dimensionality $\lambda = 1, \ldots, 9$ and noise, where the attribute values are in the range of 0.0 to 1.0. The parameters for ERiC were set to $k = 50, \mu = 2,500, \alpha = 0.999, \Delta = 0.01$. Again, the λ parameter of 4C was set to the maximal occurring correlation dimensionality, i.e., $\lambda = 9$. The remaining parameters of 4C were set to $\mu = 2,500, \varepsilon = 0.1, \delta = 0.01$. As before, the parameter k of ORCLUS was set to the exact number of correlation clusters in the data set, and parameter l was set to the maximal occurring correlation dimensionality, i.e., $k = l = 9$. Figure 11.17 illustrates the runtime of ERiC, 4C, and ORCLUS w.r.t. the data set size. The runtime of HiCO w.r.t. the size of the data set ($k = 50, \mu = 2,500, \alpha = 0.999, \Delta = 0.01$) is far above the others and therefore omitted in the chart for clearness. ERiC clearly outperforms 4C and shows a runtime comparative to that of ORCLUS.

Additionally, the overall runtime w.r.t. the number of correlation clusters in the data set of ERiC to its competitors has been compared. For this experiment five data sets of dimensionality $d = 10$ have been created. For each data set, 10,000 objects were equally distributed over a varying number $c = 20, 30, 40, 50, 60$ of correlation clusters. The parameters for ERiC and

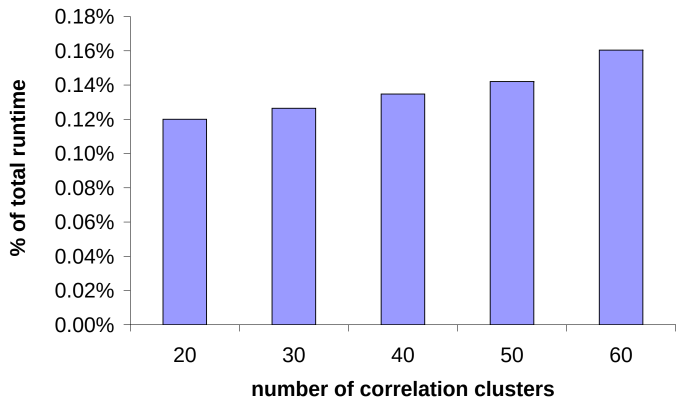


Figure 11.19: Runtime of the third step of ERiC (hierarchy aggregation) w.r.t. the number of clusters.

HiCO were set to $k = 50, \mu = 50, \alpha = 0.999, \Delta = 0.01$. 4C has been applied to the data sets with a parameter setting of $\lambda = 9, \mu = 50, \varepsilon = 0.01$ and $\delta = 0.01$. Again, the parameter k of ORCLUS was set to the exact number of correlation clusters in the data set, and parameter l was set to the maximal occurring correlation dimensionality, i.e., $l = 9$. As it can be seen in Figure 11.18 the runtime of ERiC, HiCO and 4C is quite robust w.r.t. the number of correlation clusters in the data set, while the runtime of ORCLUS increases considerably. Again, ERiC gains a significant speed-up over its competitors.

In the last efficiency experiment, the impact of the number of correlation clusters in the data set to the runtime of the third step of the ERiC algorithm, the hierarchy aggregation, has been analyzed. For this purpose, the data set of the former experiment has been used and the parameters of ERiC were also chosen as before. Figure 11.19 shows the fraction of the runtime of the third step of ERiC in comparison to the overall runtime of ERiC. As already mentioned in Section 11.2, the runtime of the hierarchy aggregation is negligible since it only requires a marginal runtime of at most 0.15% in relation to the overall runtime of the ERiC algorithm.

Part IV

Deriving Quantitative Models for Generalized Subspace Clusters

Chapter 12

Introduction

As mentioned before, the detection of correlations between different features in a given data set is a very important data mining task. High correlation of features may result in a high degree of collinearity or even a perfect one. Thus, strong correlations between different features correspond to approximate linear dependencies between two or more attributes. These dependencies can be arbitrarily complex, one or more features might depend on a combination of several other features. In the data space, dependencies of features are manifested as lines, planes, or, generally speaking, hyperplanes exhibiting a relatively high density of data points compared to the surrounding space. Knowledge concerning these arbitrary correlations is traditionally used to reduce the dimensionality of the data set by eliminating redundant features. However, detection of correlated features may also help to reveal hidden causalities that are of great importance and interest to the domain expert.

Recently, generalized subspace clustering, also called correlation clustering, has been introduced as a novel concept of knowledge discovery in databases to address the task of detection of dependencies among features and to cluster those points that share a common pattern of dependencies (cf. Chapter 9 for a detailed discussion). Again, for the sake of clarity, the expressions generalized subspace cluster/clustering and correlation cluster/clustering are used as synonyms.

Correlation clustering has been successfully applied to several application domains, e.g., see [AY00, YWWY02, BKKZ04]. For example, customer recommendation systems are important tools for target marketing. For the purpose of data analysis for recommendation systems, it is important to find homogeneous groups of users with similar ratings in subsets of the attributes. In addition, it is interesting to find groups of users with correlated affinities. This knowledge can help companies to predict customer behavior and thus develop future marketing plans. In molecular biology, correlation clustering is an important method for the analysis of several types of data. For example, in metabolic screening, the collected data set usually contains the concentrations of certain metabolites in the blood of thousands of patients. In such data sets, it is important to find homogeneous groups of patients with correlated metabolite concentrations, indicating a common metabolic disease. Thus, several metabolites can be linearly dependent on several other metabolites. Uncovering these patterns and extracting the dependencies of these clusters is a key step towards understanding metabolic or genetic disorders and designing individual drugs. A second example where generalized subspace clustering is a sound methodology for data analysis in molecular biology is DNA microarray data analysis. Microarray data usually contain the expression levels of thousands of genes expressed in different samples such as experimental conditions, cells or organisms. Roughly speaking, the expression level of a gene indicates how active this gene is, i.e., it allows the user to draw some conclusions about the amount of the product of a given gene in the given sample. The recovering of dependencies among different genes in certain conditions is an important step towards a more comprehensive understanding of the functionality of organisms which is a prominent aspect of systems biology. In addition, when the samples represent some patients, it is important to detect homogeneous groups of persons exhibiting a common linear dependency among a subset of genes in order to determine potential pathological subtypes of diseases and to develop individual treatments.

In all these cases, however, knowing merely of the existence of correlations among some features is just a first step. It is far more important to reveal quantitatively and as exactly as possible which features contribute to which dependencies as a second step. Having performed this second step, modeling a system becomes possible that describes the respective underlying data quantitatively as well as qualitatively. Thus, in order to gain the full practical potentials from generalized subspace cluster analysis, this second step is urgently needed. All existing approaches for generalized subspace clustering usually focus only on the first step of detecting the clusters. There is no method known for the second step of extracting quantitative correlation cluster information.

In this Part an approach is described to handle this second step of data analysis. General concepts are introduced for extracting quantitative information on the linear dependencies within a generalized subspace cluster such that domain experts are able to understand the correlations and dependencies in their data. In fact, the method can be applied to any correlation clusters, regardless which correlation clustering algorithm produced the results. As output, a set of linear equations is obtained that are displayed to the user. These equations can be used to understand the dependencies hidden in the analyzed data set and to create complex real-life models. As an example how this information can be used for further analysis, additionally a framework is introduced to predict the probability that a new object is generated by a specific model of the derived ones.

The remainder of this Part is organized as follows. Chapter 13 reviews related work on existing approaches for deriving descriptions of quantitative dependencies among several attributes. For a detailed discussion of related work on correlation clustering please refer to Chapter 9. Chapter 14 formalizes the notion of PCA-based correlation clusters. The concepts to derive quantitative models of correlation clusters are proposed in Chapter 15. Chapter 16 presents a broad experimental evaluation where the practical importance of the new approach is demonstrated. The concepts described in this Part have been published in [ABK+06b].

Chapter 13

Related Work

13.1 Quantitative Association Rules

An interesting approach to derive descriptive models of quantitative relationships among subsets of attributes is known as quantitative association rule mining. Some earlier approaches to this task loose information requiring discretization of attributes, e.g., [SA96]), or representation of numerical values in a rule's right-hand side by some statistical characterizations, e.g., the mean or sum of the values (cf. [Web01]). Moreover, discretization of attributes does not overcome the restriction to axis parallel dependencies. Recently, Rückert et al. [RRK04] proposed to base quantitative association rules on half-spaces, thus allowing the discovery of non-axis-parallel rules and possibly accounting for cumulative effects of several variables. The rules derived by this approach are of the form "if the weighted sum of some variables is greater than a threshold, then a different weighted sum of variables is with high probability greater than a second threshold". This approach has been shown to be useful in detecting some rules of gene-expression data sets [GRRK05]. However, these association rules do not yet uncover continuous linear dependencies, but stick to certain thresholds, reflecting the boundaries of half-spaces.

13.2 Regression Analysis

A task very similar to the one tackled in this Part is linear and multiple regression analysis (cf. [HK01] for details). The general purpose of linear regression is to learn a linear relationship between a "predictor" variable and a "response" variable. Multiple regression extends this task by allowing multiple "predictor" variables. In the following, the multiple regression model will briefly reviewed.

In the multiple linear regression model, the response variable (also referred to as the dependent variable) is assumed to be a linear function of p predictor variables (also referred to as the independent variables) plus an error introduced to account for all other factors. Formally, the model for multiple linear regression, given n observations, is for $i = 1 \dots n$:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + e_i.$$

The goal of multiple linear regression analysis is to obtain estimates of the unknown parameters $\beta_1, ..., \beta_p$ which indicate how a change in one of the independent variables affects the values taken by the dependent variable. The multiple regression equation can be rewritten more concisely in matrix notation as

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{E},$$

where $\mathbf{Y}$ is a $(n \times 1)$ data matrix (vector) containing the response variable, $\mathbf{X}$ is a $(n \times p)$ data matrix containing the observations of the p predictor variables, β is a $(p \times 1)$ data matrix containing the unknown model parameters, and $\mathbf{E}$ is a $(n \times 1)$ data matrix (vector) containing the error terms $e_1, \dots e_n$.

The usual method of estimation for the multiple linear regression model is ordinary least squares (OLS). The basic idea of OLS estimation is to choose estimates for β that minimize the sum of squared residuals $\sum_{i=1}^{n} e_i^2$. The estimated values of the parameters $\beta_1, ..., \beta_p$ are then given as

$$\hat{\beta} = (X'X)^{-1}X'y$$

Other non-linear regression models can be used to learn non-linear relationships among the predictor and the response variables. However, the main difference between regression analysis and this approach is, that in regression analysis the predictor variables are assumed to be independent. Since correlation clusters are defined to consist of points that exhibit a linear dependency among a set of attributes, the aim of this approach is to identify these dependencies when deriving a quantitative model for each cluster. Obviously, the independent variable(s) cannot be defined in advance, i.e., a set of predictor variables cannot be derived. Thus, regression analysis fails to derive quantitative models for correlation clusters as envisioned in this Part.

Chapter 14

Formalization of Correlation Clusters

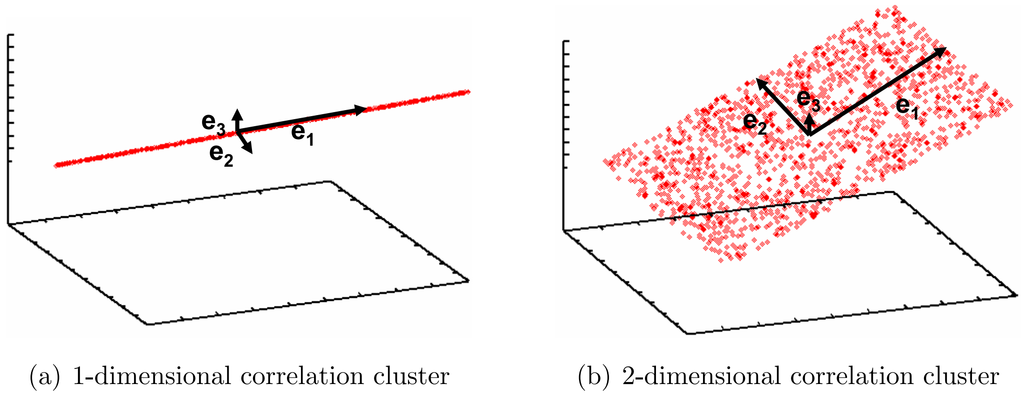
In this Chapter, the notion of correlation clusters is formalized. In the following $\mathcal{D}$ is assumed to be a database of n feature vectors in a d-dimensional real-valued feature space, i.e., $\mathcal{D} \subseteq \mathbb{R}^d$. A correlation cluster $\mathcal{C}$ is a subset of those feature vectors that are close to a common, arbitrarily oriented subspace of a given dimensionality d_i $(1 \leq d_i y d)$. In the data space the correlation clusters appear as a hyperplane of dimensionality d_i.

Generally, one way to formalize the concept of correlation clusters is to use PCA. Thus, the covariance matrix of a correlation cluster is defined as follows:

Definition 14.1 (covariance matrix of a correlation cluster).
Let $\mathcal{C} \subseteq \mathcal{D}$ be a correlation cluster that is derived using any algorithm capable of finding correlation clusters and $\bar{x}_\mathcal{C}$ denote the centroid (mean) of all points $x \in \mathcal{C}$. The covariance matrix $\mathbf{\Sigma}_\mathcal{C}$ of correlation cluster $\mathcal{C}$ *is defined as*

$$\mathbf{\Sigma}_\mathcal{C} = \frac{1}{|\mathcal{C}|} \cdot \sum_{x \in \mathcal{C}} (x - \bar{x}_\mathcal{C}) \cdot (x - \bar{x}_\mathcal{C})^T.$$

Since the covariance matrix $\mathbf{\Sigma}_\mathcal{C}$ of $\mathcal{C}$ is a positive semi-definite square matrix, it can be decomposed into the *Eigenvalue matrix* $\mathbf{E}_\mathcal{C}$ of $\mathbf{\Sigma}_\mathcal{C}$ and the



(a) 1-dimensional correlation cluster (b) 2-dimensional correlation cluster

Figure 14.1: Correlation dimensionality of correlation clusters.

Eigenvector matrix $\mathbf{V}_\mathcal{C}$ of $\mathbf{\Sigma}_\mathcal{C}$ such that

$$\mathbf{\Sigma}_\mathcal{C} = \mathbf{V}_\mathcal{C} \cdot \mathbf{E}_\mathcal{C} \cdot \mathbf{V}_\mathcal{C}^\mathbf{T}.$$

The Eigenvalue matrix $\mathbf{E}_\mathcal{C}$ is a diagonal matrix holding the d non-negative Eigenvalues of $\mathbf{\Sigma}_\mathcal{C}$ in decreasing order in its diagonal elements. The Eigenvector matrix $\mathbf{V}_\mathcal{C}$ is an orthonormal matrix with the corresponding d Eigenvectors of $\mathbf{\Sigma}_\mathcal{C}$.

Now the correlation dimensionality of $\mathcal{C}$ will be defined as the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes in $\mathbf{V}_\mathcal{C}$. Note that the correlation dimensionality is closely related to the intrinsic dimensionality of the data distribution. If, for instance, the points in $\mathcal{C}$ are located near a common line, the correlation dimensionality of these points will be 1. This means that the principal components (Eigenvectors) of the points in $\mathcal{C}$ have to be determined. The Eigenvector associated with the largest Eigenvalue has the same direction as the first principal component. The Eigenvector associated with the second largest Eigenvalue determines the direction of the second principal component and so on. The sum of the Eigenvalues equals the trace of the square matrix $\mathbf{\Sigma}_\mathcal{C}$ which is the total variance of the points in $\mathcal{C}$. Thus, the obtained Eigenvalues are equal to the variance explained by each of the principal components in decreasing order of importance. The correlation dimensionality of a set of points $\mathcal{C}$ is now defined as the smallest number of Eigenvectors explaining a portion of at least α of the total variance of $\mathcal{C}$. These ideas are illustrated in Figure 14.1. Figure 14.1(a) shows a correlation cluster of correlation dimensionality 1 corresponding to a (perfect) line. Only one Eigenvector (e_1) explains the total variance of $\mathcal{C}$. Figure 14.1(b) shows a correlation cluster of correlation dimensionality 2 that corresponds to a (perfect) plane. Here, two Eigenvectors explain the total variance of $\mathcal{C}$. Note that in the displayed examples, the correlations are perfect, i.e., there is no deviation from the hyperplane but all points within the set perfectly fit to the hyperplane. However, in real-world data sets, this is a quite unrealistic scenario. A threshold α may account for that fuzziness to define an adequate dimensionality of the correlation hyperplane. The dimensionality of a hyperplane neglecting a certain amount of deviation in orthogonal direction is called *correlation dimensionality*. The correlation dimensionality is defined more formally in the following.

Definition 14.2 (correlation dimensionality of a correlation cluster). *Let* $\alpha \in]0,1[$*, and* $\mathcal{C} \subseteq \mathcal{D}$ *be a correlation cluster. Then the* correlation dimensionality $\lambda_{\mathcal{C}}$ of $\mathcal{C}$ *is the smallest number* r *of Eigenvalues* e_i *in the* $d \times d$ *Eigenvalue matrix* $\mathbf{E}_{\mathcal{C}}$ *explaining a portion of at least* α *of the total variance:*

$$\lambda_{\mathcal{C}} = \min_{r \in \{1,\ldots,d\}} \left\{ r \;\middle|\; \frac{\sum_{i=1}^{r} e_i}{\sum_{i=1}^{d} e_i} \geq \alpha \right\}.$$

Typically, values for α are chosen between 0.8 and 0.9. for example, $\alpha = 0.85$ denotes that the obtained principal components explain 85% of the total variance. In the following, we denote the $\lambda_{\mathcal{C}}$-dimensional affine space which is spanned by the major axes of $\mathcal{C}$, i.e., by the $\lambda_{\mathcal{C}}$ first Eigenvectors of $\mathcal{C}$ and translated by, e.g., the mean vector $\bar{x}_{\mathcal{C}}$, the *correlation hyperplane* of $\mathcal{C}$.

Thus, the correlation dimensionality $\lambda_{\mathcal{C}}$ is the dimensionality of the affine space containing all points of the correlation cluster $\mathcal{C}$, allowing a small deviation corresponding to the remaining portion of variance of $1 - \alpha$. The remaining, neglected variance scatters along the Eigenvectors $e_{\lambda_{\mathcal{C}}+1}, \ldots, e_d$. Therefore, two disjoint sets of Eigenvectors will be distinguished:

Definition 14.3 (strong and weak eigenvectors of a correlation cluster).
Let $\mathcal{C} \subseteq \mathcal{D}$ be a correlation cluster, $\lambda_{\mathcal{C}}$ be the correlation dimensionality of $\mathcal{C}$, and let $\mathbf{V}_{\mathcal{C}}$ be the corresponding Eigenvectors of correlation cluster $\mathcal{C}$. The first $\lambda_{\mathcal{C}}$ Eigenvectors of $\mathbf{V}_{\mathcal{C}}$ are called strong Eigenvectors*. The strong Eigenvectors of $\mathbf{V}_{\mathcal{C}}$ are denoted by $\check{\mathbf{V}}_{\mathcal{C}}$. The remaining Eigenvectors are called* weak Eigenvectors*. The weak Eigenvectors are denoted by $\hat{\mathbf{V}}_{\mathcal{C}}$.*

For an illustration see again Figure 14.1: In the correlation cluster of correlation dimensionality 1 (Figure 14.1(a)) e_1 is a *strong Eigenvector* whereas e_2 and e_3 are *weak Eigenvectors*. In the correlation cluster of correlation dimensionality 2 (Figure 14.1(b)) e_1 and e_2 are *strong Eigenvectors* whereas e_3 is a *weak Eigenvector*. The Eigenvectors are overemphasized in this example. Suppose they were scaled by their corresponding Eigenvalues. If no variance remains along an Eigenvector as it may appear for e_2 and e_3 in Figure 14.1(a), this Eigenvector will disappear since the corresponding Eigenvalue becomes zero.

While the correlation hyperplane is spanned by the *strong* Eigenvectors, it is equally well defined by the *weak* Eigenvectors that are orthogonal to this hyperplane in $\mathbb{R}^d$. Furthermore, describing the correlation cluster by means of the weak Eigenvectors (instead of the strong Eigenvectors) directly yields an equality system that defines not only the corresponding hyperplane, but also allows the user to directly inspect the underlying dependencies among attributes numerically, as it will be shown in more detail subsequently.

Chapter 15

Deriving Quantitative Models

15.1 Deriving Correlation Cluster Models

Let $\mathcal{C}$ be a λ-dimensional correlation cluster in $\mathcal{D}$ ($\mathcal{C} \subseteq \mathcal{D}$). Thus, there are λ strong Eigenvectors and $d - \lambda$ weak Eigenvectors in the describing matrix of Eigenvectors derived by PCA on the points of cluster $\mathcal{C}$. A λ-dimensional hyperplane defining the correlation cluster $\mathcal{C}$ is therefore completely defined by the mean point (centroid) $\bar{x}_\mathcal{C} = (\bar{x}_1 \cdots \bar{x}_d)^T$ of all points belonging to cluster $\mathcal{C}$ and the set of weak Eigenvectors, $\hat{\mathbf{V}}_\mathcal{C}$, that are normal vectors to the hyperplane. Then the following equation system to describe the hyperplane can be derived, consisting of $d - \lambda$ equations:

$$\begin{array}{ccccccc}
v_{(\lambda+1),1}(x_1 - \bar{x}_1) & + v_{(\lambda+1),2}(x_2 - \bar{x}_2) & + \cdots & + v_{(\lambda+1),d}(x_d - \bar{x}_d) & = 0 \\
v_{(\lambda+2),1}(x_1 - \bar{x}_1) & + v_{(\lambda+2),2}(x_2 - \bar{x}_2) & + \cdots & + v_{(\lambda+2),d}(x_d - \bar{x}_d) & = 0 \\
 & & \vdots & & \\
v_{d,1}(x_1 - \bar{x}_1) & + \quad v_{d,2}(x_2 - \bar{x}_2) & + \cdots + & v_{d,d}(x_d - \bar{x}_d) & = 0
\end{array}$$

where $v_{i,j}$ is the value at column i, row j in the Eigenvector matrix $\mathbf{V}_\mathcal{C}$ of $\mathcal{C}$. As pointed out, only the weak Eigenvectors are relevant. Thus this equation system can be equivalently denoted by

$$\hat{\mathbf{V}}_\mathcal{C}^T \cdot x = \hat{\mathbf{V}}_\mathcal{C}^T \cdot \bar{x}_\mathcal{C}.$$

The defect of $\hat{\mathbf{V}}_{\mathcal{C}}^T$ gives the number of free attributes, the other attributes may actually be involved in linear dependencies. Basically, these dependencies are revealed by transforming the equation system using Gauss-Jordan elimination. The thus derived reduced row echelon form of the matrix is known to be unique [Yus84]. The unique form does, of course, not provide new information, but it is easily comparable to alternative solutions and conveniently interpretable by inspecting experts. To enhance numerical stability, the use of total pivoting for the Gauss-Jordan elimination is supposed.

By construction, the equation system is – at least approximately – fulfilled for all points $x \in \mathcal{C}$. But, furthermore, it suggests a quantitative model for the cluster. This model could be evaluated by using retained data points. Besides, as shown below, it may also serve as a predictive model to classify new data points.

In summary, the following general method to derive quantitative models of clusters in a data set of feature vectors $\mathcal{D} \subset \mathbb{R}^d$ is proposed:

1. Run a clustering algorithm on $\mathcal{D}$ that is able to find correlation clusters, i.e., use, e.g., 4C [BKKZ04] or ORCLUS [AY00]. However, also k-means [McQ67] or DBSCAN [EKSX96] is possible, provided that a proper distance function is used which takes the correlation dimension into account. If the result may be restricted to clusters of *positively* correlated features, even the usage of any general biclustering [Har72] or pattern-based clustering algorithm [WWYY02, YWWY02, LW03, PZC+03] will be possible. The decision for a specific clustering algorithm will also determine whether or not a data object may belong to several clusters simultaneously. In the experiments, the algorithm COPAC [ABK+07c] is applied, a correlation clustering algorithm that is shown to improve over 4C as well as over ORCLUS w.r.t. efficiency, effectivity, and robustness.

2. For each correlation cluster $\mathcal{C}_i \subset \mathcal{D}$ found in the previous step:

    (a) Derive the covariance matrix $\mathbf{\Sigma}_{\mathcal{C}_\mathrm{i}}$.

    (b) Select the weak Eigenvectors $\hat{\mathbf{V}}_{\mathcal{C}_\mathrm{i}}$ of $\mathbf{\Sigma}_{\mathcal{C}_\mathrm{i}}$ with respect to a certain threshold α.

(c) Derive the equation system describing the correlation hyperplane:

$$\hat{\mathbf{V}}_{\mathcal{C}_\mathbf{i}}^T \cdot x = \hat{\mathbf{V}}_{\mathcal{C}_\mathbf{i}}^T \cdot \bar{x}_{\mathcal{C}_i}$$

(d) Apply Gauss-Jordan elimination to the derived equation system to obtain a unique description of quantitative dependencies by means of the reduced row echelon form of the equation system.

15.2 Interpretation of Correlation Cluster Models

Suppose by applying this method, the following solution describing a cluster in a 5-dimensional feature space $\mathbb{R}^5$ is obtained:

$$\begin{aligned}
1x_1 + 0x_2 + c_1x_3 + 0x_4 + e_1x_5 &= f_1\\
0x_1 + 1x_2 + c_2x_3 + 0x_4 + e_2x_5 &= f_2\\
0x_1 + 0x_2 + \ 0x_3 \ + 1x_4 + e_3x_5 &= f_3
\end{aligned}$$

This would provide a quantitative model describing a correlation cluster of correlation dimensionality 2 (corresponding to the number of free attributes, or, equivalently, the number of *strong* Eigenvectors) where linear dependencies exist among

- x_1, x_3, and x_5
- x_2, x_3, and x_5
- x_4 and x_5

by given factors c_1, e_1, c_2, e_2, and e_3.

Note that there must not be drawn any conclusions concerning causalities between attributes. But relations between certain attributes are quantitatively and uniquely defined. To resolve these relations to any formula that suggests a causality, one has to rely on the experts domain knowledge. However, uncovered quantitative relationships will lead to refined experiments and help to finally explore supposable causalities. Thus, experimental settings could be chosen involving either

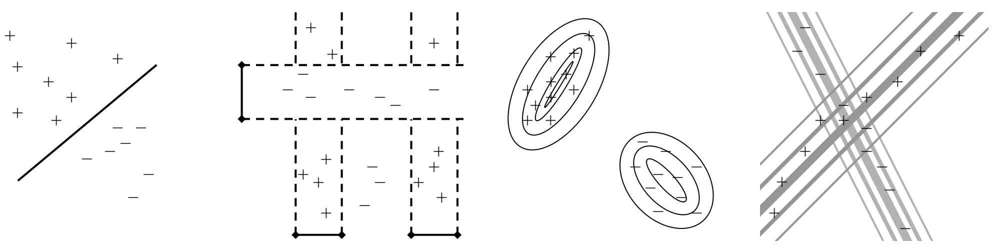
- x_4 and x_5 or
- x_2, x_3, and x_5 or
- x_1, x_3, and x_5,

and changing the quantities in relation to each other. The dependencies revealed in the original experiment could have been interpreted such as fall or rise of an arbitrary subset of $S \subset \{x_1, x_3, x_5\}$ that caused fall or rise of the remaining subset $\{x_1, x_3, x_5\} \setminus S$. Further experiments could refine the model by excluding certain combinations of causal models. Of course, the three variables, x_1, x_3, and x_5, may also simply be connected by a fourth variable that has not been monitored so far. Thus, trivially, a quantitative connection will never guarantee a direct causal relationship. Furthermore, in many domains, one-way causal relationships provide only one part of the whole picture, since systems often are regulated by negative-feedback-loops that make causalities circular. Nevertheless, modeling parts of a complex system remains useful even under restrictive constraints, e.g., as shown for genetic regulatory interaction networks in [Hus03].

15.3 Sample Application: Predictive Models

Having derived a descriptive model, it can be refined by determining an average distance of the cluster members from the correlation hyperplane. Such deviations are typically expected in natural systems. At least, one has to account for errors in measurement. The distance of a point to a hyperplane is thereby naturally defined as the Euclidean distance to its perpendicular projection onto the hyperplane, i.e.,

$$d(x, \mathcal{C}) = ||x - \bar{x}_{\mathcal{C}} - \mathrm{proj}_{\mathcal{C}-\bar{x}_{\mathcal{C}}}(x - \bar{x}_{\mathcal{C}})||,$$



(a) Linear decision boundaries (b) Axis parallel decision rules (c) Density functions (d) Deviations from hyperplanes

Figure 15.1: Decision models of different types of classifiers.

where $\mathcal{C}$ denotes the idealized hyperplane of a correlation cluster. By definition, the hyperplane $\mathcal{C}$ is an affine space, i.e., a subspace translated by $\bar{x}_{\mathcal{C}}$, the mean vector of all points of the cluster corresponding to $\mathcal{C}$. The function $\text{proj}_S : \mathbb{R}^n \to \mathbb{R}^n$ denotes the perpendicular projection of a vector to an arbitrary subspace S of $\mathbb{R}^n$. If S is given by an orthonormal basis, e.g., the set of strong Eigenvectors derived for the corresponding correlation cluster, $\{s_1, \cdots, s_{\lambda_S}\}$, then

$$\text{proj}_S(x) = \langle x, s_1 \rangle s_1 + \langle x, s_2 \rangle s_2 + \cdots + \langle x, s_{\lambda_S} \rangle s_{\lambda_S}.$$

Assuming the deviations fit to a Gaussian distribution with $\mu = 0$, the standard deviation σ of the distances of all cluster members suffices to define a Gaussian model of deviations from the common correlation hyperplane. For each of the derived models, the probability is given for a new data object to be generated by this specific Gaussian distribution. A set of models for a set of correlation clusters can therefore provide a convenient instrument for classification in the perspective of different linear dependencies among the data. The probability that an object x was generated by the jth of n Gaussian distributions, $\mathcal{C}_j$, is given by

$$P(\mathcal{C}_j|x) = \frac{\frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2\sigma_j^2}(d(x,\mathcal{C}_j))^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2\sigma_i^2}(d(x,\mathcal{C}_i))^2}}.$$

Compared to many traditional classification algorithms, like SVM or kNN, these predictive models do not only provide a separating boundary between classes (cf. Figure 15.1(a)), but also give a meaningful definition of the class. So do other classifiers, like decision trees or rule based learners, but their descriptions usually are limited to (at least in sections) axis parallel decision boundaries (cf. Figure 15.1(b)). The models provided by the EM algorithm or other Bayesian learners differ from the proposed models in that they simply define a scattering around a mean point using a quadratic form distance function or a density function for a certain probability distribution (cf. Figure 15.1(c)). For underlying linear dependencies, a quadratic distance function will resemble these new models only if the dependencies are perfectly expressed in the data without any aberrations. Accounting for some variance perpendicular to a hyperplane, while the hyperplane represents a linear dependency among several attributes, is a novel approach among the family of classification algorithms (cf. Figure 15.1(d)).

Chapter 16

Evaluation

All experiments have been performed on a workstation with a $2 \cdot 64$-bit 2.6 GHz CPU and 16 GB main memory. All evaluated methods have been implemented in Java. In all experiments, the input parameters of all methods have been optimized in terms of quality and the best results have been reported in order to achieve a fair comparison. In all experiments the correlation clustering algorithm COPAC [ABK+07c] has been used to determine the correlation clusters in a preprocessing step. This algorithm has been chosen due to its efficiency, effectivity, and robustness. In each case, parameters for clustering were chosen according to [ABK+07c]. Again, please note that any other correlation clustering algorithm is applicable for preprocessing.

16.1 Synthetic Data Sets

For the experiments several synthetic data sets have been used containing correlation clusters in the unit cube of $\mathbb{R}^d$ that have been generated by a generic data generator. The generated correlation clusters form a λ-dimensional hyperplane which is specified by an equation system of $d - \lambda$ equations. The distances of the points to the hyperplane are normally distributed with a specified standard deviation and a mean of zero.

The first data set "DS1" consists of five correlation clusters, each forming

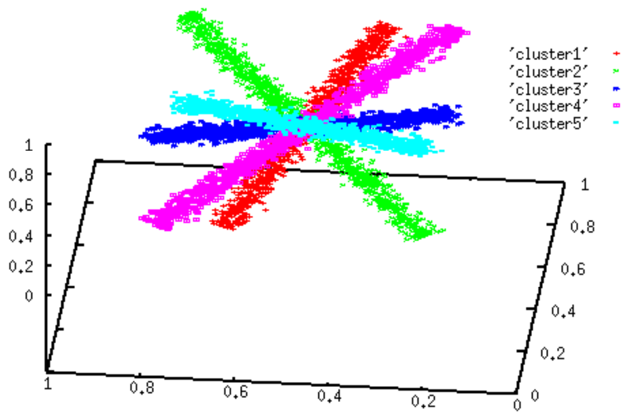


Figure 16.1: Synthetic data set "DS1".

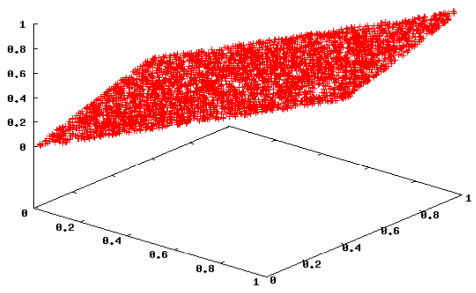
a line of 1,000 points in $\mathbb{R}^3$ (cf. Figure 16.1). In each cluster, the distances of the points to the correlation lines are normally distributed with a standard deviation of about 1.5% of the maximum distance in the unit cube. The purpose of this data set is to demonstrate the capability of the proposed method to obtain a quantitative model for the correlation clusters. As it can be seen in Table 16.1 a good approximation of the equation systems has been derived that define the models for the correlation clusters despite the obviously strong jitter in the data set.

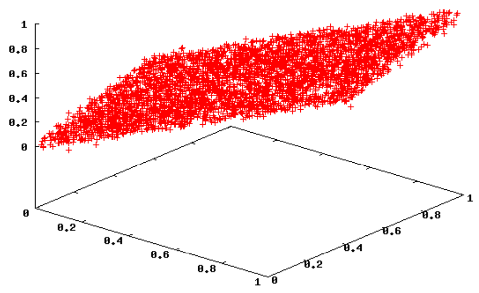
In the second experiment, the proposed method has been evaluated on data sets with varying standard deviation. For this purpose, six data sets ("DS2$_0$", ..., "DS2$_5$") have been generated, forming a 2-dimensional hyperplane in $\mathbb{R}^3$ with different values for the standard deviation of the distances. The generated dependencies follow the equation $x1 - 0.5x2 - 0.5x3 = 0$. The values for the standard deviation were set to $\sigma_0 = 0\%$ up to $\sigma_5 = 5\%$ of the maximum distance in the unit cube (cf. Figure 16.2). The results are shown in Table 16.2. As expected, with increasing standard deviation of the distances, the detected correlation models suffer from a slight blurring, i.e., the coefficients of the models slightly deviate from the exact coefficients. However, the general correlations are still detected and also the hidden quantitative relationships are still uncovered rather clear even if the points stronger deviate from the optimal hyperplane. In general, the proposed method has proven to be rather robust w.r.t. small jitter.

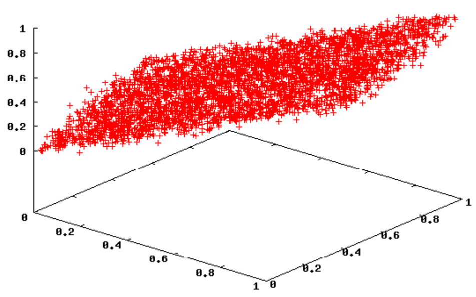
Table 16.1: Dependencies on "DS1" data.

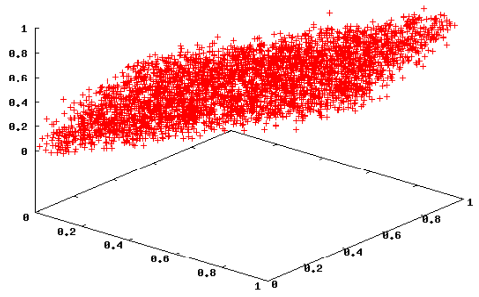
	Generated dependencies	Generated standard deviation	Found dependencies
cluster 1	$x1 - x3 = 0$ $x2 + 0.5x3 = 0.75$	$\sigma = 0.0246$	$x1 - 1.0069x3 = -0.0035$ $x2 + 0.5065x3 = 0.7537$
cluster 2	$x1 - x3 = 0$ $x2 - x3 = 0$	$\sigma = 0.0243$	$x1 - 1.0027x3 = -0.0028$ $x2 - 0.9901x3 = 0.0022$
cluster 3	$x1 + x3 = 1$ $x2 - x3 = 0$	$\sigma = 0.0238$	$x1 + 1.0008x3 = 1.0005$ $x2 - 1.0011x3 = 0.0000$
cluster 4	$x1 - x3 = 0$ $x2 + x3 = 1$	$\sigma = 0.0246$	$x1 - 1.0009x3 = 0.0000$ $x2 + 0.9999x3 = 0.9995$
cluster 5	$x1 + x3 = 1$ $x2 + x3 = 1$	$\sigma = 0.0249$	$x1 + 0.9975x3 = 0.9988$ $x2 + 0.9968x3 = 0.9992$

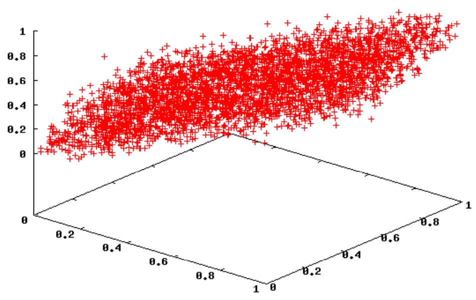
In addition to the reported experiments on 3-dimensional data, several similar experiments on higher dimensional data have been performed. In all experiments, results of similar high quality have been achieved, i.e., all linear dependencies hidden in the data were correctly uncovered.


(a) DS2_0 ($\sigma_0 = 0$)


(b) DS2_1 ($\sigma_1 = 0.0173$)


(c) DS2_2 ($\sigma_2 = 0.0346$)


(d) DS2_3 ($\sigma_3 = 0.0520$)


(e) DS2_4 ($\sigma_4 = 0.0693$)

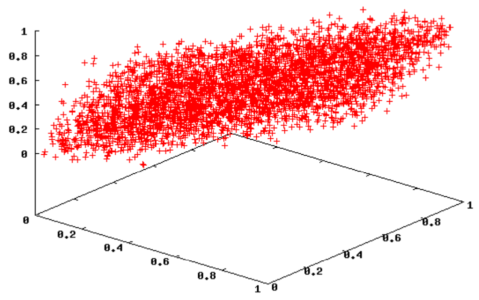

(f) DS2_5 ($\sigma_5 = 0.0866$)

Figure 16.2: Synthetic data sets "DS2" with different values for standard deviation.

Table 16.2: Found dependencies on “DS2” data, with generated dependencies: $x1 - 0.5x2 - 0.5x3 = 0$.

	Generated standard deviation	Found dependencies
“DS2$_0$”	$\sigma_0 = 0$	$x1 - 0.5000x2 - 0.5000x3 = 0.0000$
“DS2$_1$”	$\sigma_1 = 0.0173$	$x1 - 0.4989x2 - 0.5002x3 = 0.0000$
“DS2$_2$”	$\sigma_2 = 0.0346$	$x1 - 0.5017x2 - 0.4951x3 = 0.0016$
“DS2$_3$”	$\sigma_3 = 0.0520$	$x1 - 0.5030x2 - 0.5047x3 = -0.0059$
“DS2$_4$”	$\sigma_4 = 0.0693$	$x1 - 0,4962x2 - 0.5106x3 = -0.0040$
“DS2$_5$”	$\sigma_5 = 0.0866$	$x1 - 0.4980x2 - 0.4956x3 = 0.0064$

16.2 Real-world Data Sets

"Wages" data. The "Wages' data set[1] consists of 534 11-dimensional observations from the 1985 Current Population Survey. Since most of the attributes are not numeric, we used only 4 dimensions (A=age, YE=years of education, YW=years of work experience, and W=wage) for correlation analysis.

COPAC detected three correlation clusters in this data set. The resulting dependencies of these clusters are summarized in Table 16.3. The first cluster consists only of people having 12 years of education, whereas the second cluster consists only of people having 16 years of education. Furthermore, in both of these clusters the difference between age and work experience is a specific constant, namely years of education plus 6, which makes perfectly sense. Additionally, for the first cluster, a dependency between wage and age has been found: the wage equals a constant plus a small factor times the age of an employee, i.e., the older an employee, the more he earns. This relationship is independent from the attribute work experience. Note that years of education is a constant where this condition holds. In the third cluster only those employees are grouped which started school in the age of 6 years and after graduation immediately began working. Thus, the sum of years of education and work experience equals the age minus 6.

"Gene Expression" data. This data set was derived from an experimental study of apoptosis in human tumor cells[2]. Apoptosis is a genetically controlled pathway of cell death. The data set contains the expression level of 4,610 genes at five different time slots (5, 10, 15, 30, 60 minutes) after initiating the apoptosis pathway.

The two correlation clusters with cluster ID (cID) 1 and 2 detected by COPAC have been analyzed. The derived dependencies of these clusters are depicted in Table 16.4. The attributes are abbreviated by Mi, where i

[1] http://lib.stat.cmu.edu/datasets/CPS_85_Wages

[2] The data are donated by project partners.

Table 16.3: Dependencies on "Wages" data.

	derived dependencies
cluster 1	$YE = 12$
	$YW - 1 \cdot A = -18$
	$W - 0.07 \cdot A = 5.14$
cluster 2	$YE = 16$
	$YW - 1 \cdot A = -22$
cluster 3	$YE + 1 \cdot YW - 1 \cdot A = -6$

denotes the time slot of this attribute, e.g., $M5$ denotes time slot "5 minutes". The first cluster contains several genes that are located at the mitochondrial membrane. The first four time slots exhibit a negative linear relationship with $M60$. Similar observations can be made for the second cluster that contains several genes that are related to the tumor necrosis factor (RNF). The uncovered dependencies suggest that the activity of the corresponding genes decrease with proceeding cell death. The strong negative correlations among genes related to mitochondria (cluster 1) indicates that the volume of the energy metabolism (which is located in mitochondria) is decreasing over time. In addition, the correlation among the genes related to RNF makes sense since the dying cells are tumor cells.

"Breast Cancer" data. The proposed method has also been applied to four correlation clusters found in the Wisconsin Breast Cancer data derived from UCI ML Archive[3]. This data set measures nine biomedical parameters characterizing breast cancer type in 683 humans (humans with missing values were removed from the data set). The parameters include Clump Thickness (attribute "A1"), Uniformity of Cell Size ("A2"), Uniformity of Cell Shape

[3]http://www.ics.uci.edu/~mlearn/MLSummary.html

Table 16.4: Dependencies on "Gene Expression" data.

cID	derived dependencies	sample gene names
1	$M5 - 1.05 \cdot M60 = -0.12$ $M10 - M60 = -0.17$ $M15 - M60 = 0$ $M30 - 1.1 \cdot M60 = 0.11$	NDUFB10, MTRF1, TIMM17A, TOM34, CPS1, NM44, COX10, FIBP, TRAP1, MTERF, ME2, HK1, HADHA, ASAH2, CPS1, CA5A, BNI3PL
2	$M5 - 0.98 \cdot M60 = 0$ $M10 - 0.98 \cdot M60 = 0$ $M15 - 0.97 \cdot M60 = 0$ $M30 - 0.97 \cdot M60 = 0$	TNFRSF6, TNFRSF11A, TNFRSF7, TNFRSF1B, TNFRSF10B,TNFRSF5, TNFRSF1A, TRAF5, TRAF2, TNFSF12

("A3"), Marginal Adhesion ("A4"), Single Epithelial Cell Size ("A5"), Bare Nuclei ("A6"), Bland Chromatin ("A7"), Normal Nucleoli ("A8"), and Mitoses ("A9").

The derived dependencies of the four clusters with cluster ID (cID) $1, \ldots, 4$ are depicted in Table 16.5. It has to be stressed that each cluster only contains humans suffering from a benign tumor type. The patients suffering from a malignant tumor type were classified as noise. The dependencies in the first cluster are quite clean and indicate a constant behavior of seven attributes. In addition, $A5$ is related to $A7$. The models of the remaining clusters are quite complex. Mostly, the first attributes which measure an aggregated information about the shape and the size of the tumor cells exhibit a relationship to more specific measurements on single parts of the tumor. In general, since the clusters only contain benign tumors, the results indicate that this mostly harmless tumor type can still be explained and modeled by linear relationships among the measurements, whereas the more dangerous tumor type cannot be explained or modeled through any linear relations

Table 16.5: Dependencies on “Breast Cancer” data.

cID	derived dependencies
1	$A1 = 2$ and $A2 = 1$ and $A3 = 1$ and $A4 = 1$ and $A6 = 1$ and $A5 - 0.1 \cdot A7 = 1.9$ $A8 = 1$ and $A9 = 1$
2	$A1 - 0.4 \cdot A4 + 0.7 \cdot A5 - 0.2 \cdot A6 + 0.9 \cdot A7 - 24 \cdot A8 = -20.9$ $A2 + 0.03 \cdot A4 - 0.05 \cdot A5 + 0.02 \cdot A6 + 0.02 \cdot A7 - 0.3 \cdot A8 = 0.8$ $A3 + 0.2 \cdot A4 + 0.1 \cdot A5 + 0.1 \cdot A6 + 0.2 \cdot A7 - 1.8 \cdot A8 = 0.3$
3	$A1 + 82.2 \cdot A6 + 7.8 \cdot A7 - 42 \cdot A8 - 18.5 \cdot A9 = 38.5$ $A2 - 1.9 \cdot A6 - 0.2 \cdot A7 + 0.9 \cdot A8 + 1.8 \cdot A9 = 1.5$ $A3 - 60.1 \cdot A6 - 6.5 \cdot A7 + 25.1 \cdot A8 + 141 \cdot A9 = 97.5$ $A4 - 7.2 \cdot A6 - 0.4 \cdot A7 - 1.1 \cdot A8 + 15.6 \cdot A9 = 7.6$ $A5 - 18.8 \cdot A6 - 1.4 \cdot A7 - 0.5 \cdot A8 + 45.9 \cdot A9 = 26.1$
4	$A1 - 5.4 \cdot A5 + 1.6 \cdot A6 - 0.1 \cdot A7 + 1 \cdot A8 - 16.3 \cdot A9 = -21.1$ $A2 + 1.7 \cdot A5 - 0.6 \cdot A6 + 0.2 \cdot A7 - 0.7 \cdot A8 - 9.9 \cdot A9 = -6.5$ $A3 - 1.8 \cdot A5 - 0.8 \cdot A6 - 0.3 \cdot A7 - 0.7 \cdot A8 - 11.9 \cdot A9 = -8.5$ $A4 - 2.3 \cdot A5 - 0.2 \cdot A6 + 0.2 \cdot A7 + 0.4 \cdot A8 + 8.6 \cdot A9 = 6.5$

among the measurements.

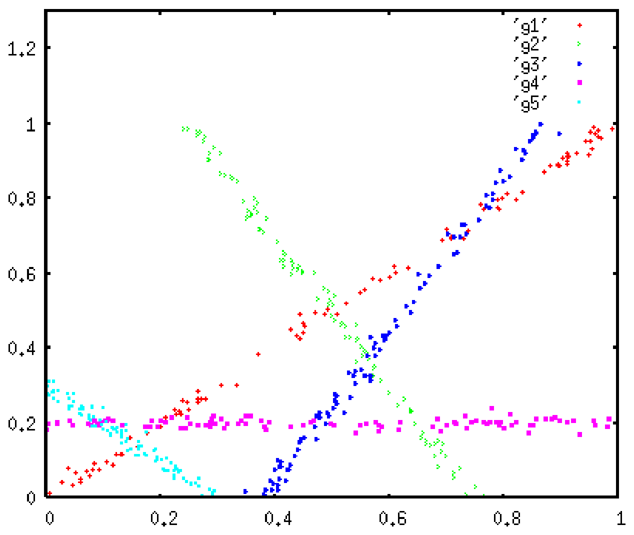


Figure 16.3: Data set "DS3$_2$".

16.3 Applying Quantitative Models to Class Prediction

Last but not least, a further potential application of the proposed method is briefly discussed that utilizes the derived models for subsequent data analysis. As sketched above, the quantitative models generated by the proposed approach can be used to predict the class of a new object. To evaluate this potential three 2-dimensional synthetic data sets each with 5 classes have been used. The first data set ("DS3$_0$") contains 50 points per class, the second and the third data sets ("DS3$_1$" and "DS3$_2$") each contain 100 points per class. Each class is generated according to a certain linear dependency. The class distributions in "DS3$_0$" and "DS3$_1$" exhibit a jitter of 0.5% of the maximum distance in the unit cube, whereas the jitter of the classes in "DS3$_2$" is 0.75%. The third data set is depicted in Figure 16.3. Note that these data sets are rather artificial and are only applied for a proof of principle.

The classification accuracy of the sketched classifier has been compared to several other standard learning approaches. For this comparison the WEKA framework [WF05] has been used with standard parameter settings, in partic-

Table 16.6: Comparison of different classifiers in terms of accuracy (in %).

	Proposed method	IBk	SMO	PART	NB	J48	Log.
“$DS3_0$”	95	91	62	82	65	82	67
“$DS3_1$”	94	94	54	85	64	83	60
“$DS3_2$”	91	91	58	81	60	83	57

ular, kNN (IBk) with $k = 1$ (best results reported), SVM (SMO), rule-based learner (PART), Naive Bayes, decision tree (J48), and multinomial logistic regression (Logistic). The results are depicted in Table 16.6. As it can be seen, the proposed approach significantly outperforms most of the other approaches, except kNN, in terms of accuracy.

Please note that standard classifiers will most likely produce comparative or even better results if the classes are generated through models that cannot be captured by the concepts of linear dependencies. However, this small example may show that if the classes are generated by a model of linear dependencies as captured by the proposed concepts, the proposed method obviously yields a better prediction accuracy than standard supervised learners.

Part V

Conclusions and Outlook

Chapter 17

Summary and Future Directions

Within the KDD process, data mining is the application of algorithms to discover patterns and trends in large databases. Clustering is one of the most important data mining tasks. The methods and concepts presented in this thesis contribute to the field of hierarchical subspace clustering. This chapter summarizes the main contributions of this thesis (cf. Section 17.1) and shows some directions for future work (cf. Section 17.2).

17.1 Summary of Contributions

The rapidly increasing amount of data stored in databases requires efficient and effective data mining methods to gain new information contained in the collected data. Clustering is one of the primary data mining tasks and aims at detecting subgroups of similar data objects. This thesis contributes to the field of hierarchical clustering. Novel challenges for the hierarchical approach to subspace clustering are identified and innovative and solid solutions for these challenges are proposed. In the following, a detailed summary of these contributions is given.

17.1.1 Preliminaries (Part I)

The preliminaries in Part I provide some motivation and illustrate the topic and the background of this work. After a very general introduction to the process of Knowledge Discovery in Databases and Data Mining, a brief overview of traditional hierarchical clustering methods is given. Afterwards, an introduction to the density-based notion of clusters is provided. In particular, the notion of density-connectivity underlying the algorithm DBSCAN [EKSX96] is introduced. Then, its hierarchical extension leading to the notion of hierarchical density-based clustering which constitutes the central concept of the algorithm OPTICS [ABKS99] is discussed.

17.1.2 Hierarchical Axis-Parallel Subspace Clustering (Part II)

Part II deals with the analysis of hierarchical subspace clusters in axis-parallel subspaces. Subspace clustering can be seen as an extension of traditional clustering which aims at automatically identifying lower dimensional axis-parallel subspaces of the feature space in which clusters exist. After discussing existing approaches on axis-parallel subspace clustering, two new algorithms are presented for detecting hierarchies of subspace clusters in axis-parallel subspaces.

First, the algorithm HiSC (Hierarchical Subspace Clustering) is proposed which is the first subspace clustering algorithm for detecting hierarchies of subspace clusters. HiSC scales linearly in the dimensionality of the data space and quadratically in the number of points. It is superior to the state-of-the-art subspace clustering algorithms in several aspects: First, HiSC can detect clusters in subspaces of significantly different dimensionality. Second, HiSC is able to determine hierarchies of nested subspace clusters, i.e., the relationships of lower dimensional subspace clusters that are embedded within higher dimensional subspace clusters. Third, HiSC does not rely on a global clustering criterion and, thus, is able to detect clusters of different size, shape, and density. Fourth, the choice of parameters is considerably simplified compared to previous methods. Several comparative experiments using synthetic and real-world data sets show that HiSC has a superior performance and effectivity compared to existing methods.

Additionally, the algorithm DiSH (Detecting Subspace Cluster Hierarchies) is introduced which is a major extension of HiSC. While HiSC is limited to single inclusions of subspace cluster hierarchies, DiSH is able to determine hierarchies of single and multiple inclusions. DiSH applies a density-based hierarchical approach similar to OPTICS [ABKS99] and, thus, avoids Single-Link effects like they can occur while using HiSC. Furthermore, DiSH computes a clear and intuitive graph representation of the result such that the complete hierarchical relationships among subspace clusters can be seen at a glance. The complete runtime complexity of DiSH results in $O(n^2 \cdot d^2)$, where n is the number of objects and d is the dimensionality of the data space. Applying DiSH to synthetic and real-world data sets yields further important insights of the hierarchical subspace clustering structure that have been missed by HiSC.

17.1.3 Hierarchical Subspace Clustering in Arbitrarily Oriented Subspace (Part III)

Part III discusses new methods for hierarchical clustering in arbitrarily oriented subspaces of the feature space. The so-called correlation clustering can be seen as an extension of axis-parallel subspace clustering. Correlation clustering aims at grouping the data set into subsets, the so-called correlation clusters, such that the objects in the same correlation cluster show uniform attribute correlations. In the first Chapter existing work on the research area of correlation clustering is reviewed and discussed. Then, two new algorithms are introduced for identifying hierarchies of correlation clusters in arbitrarily oriented subspaces.

The algorithm HiCO (Hierarchical Correlation Ordering) is the first algorithm for computing hierarchies of correlation clusters, i.e., lower dimensional correlation clusters which are embedded within larger dimensional correlation clusters. In contrast to existing approaches, this method does not require the user to specify any global density threshold, the number of clusters to be found, nor any parameter specifying the dimensionality of the correlations. The complete runtime complexity of HiCO yields $O(n^2 \cdot d^3)$, with n being the number of objects and d being the dimensionality of the data space. The experimental evaluation shows that HiCO finds meaningful and rich hierarchies of correlation clusters in synthetic and real-world data sets.

Afterwards, the search for complex hierarchies of correlation clusters including the information that lower dimensional correlation clusters are embedded within multiple higher dimensional ones is motivated. Since none of the existing algorithms for correlation clustering can reveal the complete hierarchical structure, the algorithm ERiC (Exploring complex hierarchical Relationships among Correlation clusters) is introduced, a novel clustering algorithm to detect complex hierarchical relationships between correlation clusters also allowing for multiple inclusions. The resulting cluster hierarchy is visualized by means of a clear graph model. The complete runtime of ERiC results in $O(n^2 \cdot d^2)$, where n is the number of objects and d denotes the dimensionality of the data space. It is shown experimentally that ERiC outperforms existing state-of-the-art correlation clustering algorithms in terms of runtime and accuracy.

17.1.4 Deriving Quantitative Models for Generalized Subspace Clusters (Part IV)

None of the existing correlation clustering algorithms derives a quantitative model for each correlation cluster which is urgently needed in order to gain the full practical potentials from correlation cluster analysis. Part IV describes an original approach to derive quantitative information on the linear dependencies within correlation clusters. The concepts are independent of the clustering model and can thus be applied as a post-processing step to any correlation clustering algorithm. Furthermore, as a sample application of the approach, it is sketched how these quantitative models can be used to predict the probability distribution that an object is created by these models.

The broad experimental evaluation demonstrates the beneficial impact of the proposed method on several applications of significant practical importance. It is exemplified how the method can be used in conjunction with a suitable clustering algorithm to gain valuable and important knowledge about complex relationships in real-world data.

17.2 Potentials for Future Work

At the end of this thesis, the potentials of the proposed methods for future research are emphasized. For generalized subspace clustering of high-dimensional data, future research could be guided in the following directions:

- While much work has been done in identifying linear correlation among subsets of features in high-dimensional data, the field of detecting non-linear correlations is quietly unexplored. Using standard (linear) PCA algorithms can detect linear correlations within a data set, but fail in identifying nonlinear structures. The concept of kernel PCA is a new method for applying a nonlinear form of PCA, and thus, is very well suited to extract nonlinear structures in the data. It would be interesting to investigate how the concepts of Kernel PCA could be combined with the notion of density-based clustering to find nonlinear correlation clusters in arbitrarily oriented subspaces.
- Another interesting idea would be to extend the derivation of a quantitative model for linear correlation clusters to one for nonlinear correlation clusters. This problem may be solved by applying the so-called kernel trick to map the original observations into a higher dimensional feature space. Then, one could try to find a quantitative model of the correlation cluster in the feature space and perform a transformation of the derived model back to the original data space.
- Beside PCA (used in HiCO and ERiC as proposed in this thesis), there are several other concepts, such as the Hough transform or fractal dimension which could be used for correlation clustering. For instance,

the Hough transform maps the data space to a parameter space, defining the set of possible arbitrarily oriented subspaces. An interesting approach would be to integrate the principles of the Hough transform into a clustering algorithm, i.e., to find those among all the possible subspaces that accommodate many database objects. This would lead to a clustering algorithm which is independent of any distance measure and is able to subspace clusters which are sparse and/or intersected by other clusters within a noisy environment.

List of Figures

List of Tables

Bibliography

[ABK$^+$06a] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Finding hierarchies of subspace clusters. In *Proceedings of the 10th European Conference on Principles of Knowledge Discovery and Data Mining (PKDD), Berlin, Germany*, 2006.

[ABK$^+$06b] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. Deriving quantitative models for correlation clusters. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA*, 2006.

[ABK$^+$07a] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Detection and visualization of subspace cluster hierarchies. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA), Bangkok, Thailand*, 2007.

[ABK$^+$07b] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. On exploring complex relationships of correlation clusters. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada*, 2007.

[ABK$^+$07c] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. Robust, complete, and efficient correlation clustering. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM), Minneapolis, MN*, 2007.

[ABKS99] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the SIGMOD Conference, Philadelphia, PA*, 1999.

[ABKZ06] E. Achtert, C. Böhm, P. Kröger, and A. Zimek. Mining hierarchies of correlation clusters. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM), Vienna, Austria*, 2006.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the SIGMOD Conference, Seattle, WA*, 1998.

[APW$^+$99] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the SIGMOD Conference, Philadelphia, PA*, 1999.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the SIGMOD Conference, Minneapolis, MN*, 1994.

[AY00] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional space. In *Proceedings of the SIGMOD Conference, Dallas, TX*, 2000.

[BKKK04] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, U.K.*, 2004.

[BKKZ04] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *Proceedings of the SIGMOD Conference, Paris, France*, 2004.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference Intelligent Systems for Molecular Biology (ISMB), San Diego, CA*, 2000.

[CFZ99] C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA*, pages 84–93, 1999.

[Def77] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–31, 1977.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR*, 1996.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR*, 1996.

[GHPT05] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas. Dimension induced clustering. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL*, 2005.

[GNC99] S. Goil, H. Nagesh, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. Technical Report CPDC-TR-9906-010, Northwestern University, 1999.

[GRRK05] E. Georgii, L. Richter, U. Rückert, and S. Kramer. Analyzing microarray data using quantitative association rules. *Bioinformatics*, 21(Suppl. 2):ii1–ii8, 2005.

[Har72] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Academic Press, 2001.

[Hus03] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[KKK04] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Orlando, FL*, 2004.

[KKRW05] H.-P. Kriegel, P. Kröger, M. Renz, and S. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the 5th International Conference on Data Mining (ICDM), Houston, TX*, 2005.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analyis.* John Wiley&Sons, 1990.

[LW03] J. Liu and W. Wang. OP-Cluster: Clustering by tendency in high dimensional spaces. In *Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL*, 2003.

[McQ67] J. McQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematics, Statistics, and Probabilistics*, volume 1, pages 281–297, 1967.

[MO04] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[NH94] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Santiago de Chile, Chile*, 1994.

[PJAM02] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the SIGMOD Conference, Madison, WI*, 2002.

[PZC$^+$03] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. MaPle: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL*, 2003.

[RRK04] U. Rückert, L. Richter, and S. Kramer. Quantitative association rules based on half-spaces: an optimization approach. In *Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, U.K.*, pages 507–510, 2004.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the SIGMOD Conference, Montreal, Canada*, 1996.

[SEKX98] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2:169–194, 1998.

[Sib73] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[SSZ$^+$98] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization.". *Molecular Biolology of the Cell*, 9:3273–3297, 1998.

[TXO05] A. K. H. Tung, X. Xu, and C. B. Ooi. CURLER: Finding and visualizing nonlinear correlated clusters. In *Proceedings of the SIGMOD Conference, Baltimore, ML*, 2005.

[Voo86] E. M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing and Management*, 22(6):465–576, 1986.

[Web01] G. I. Webb. Discovering associations with numeric variables. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Francisco, CA*, pages 383–388, 2001.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[WWYY02] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the SIGMOD Conference, Madison, WI*, 2002.

[Yus84] T. Yuster. The reduced row echelon form of a matrix is unique: A simple proof. *Mathematics Magazine*, 57(2):93–94, 1984.

[YWWY02] J. Yang, W. Wang, H. Wang, and P. S. Yu. Delta-Clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose, CA*, 2002.