Correlation Clustering

Arthur Zimek

München 2008

Correlation Clustering

Arthur Zimek

Dissertation
an der Fakultät für Mathematik, Informatik und
Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Arthur Zimek

München, den 06.05.2008

Erstgutachter: Prof. Dr. Hans-Peter Kriegel,
Ludwig-Maximilians-Universität München

Zweitgutachter: Prof. Dr. Thomas Seidl,
Rheinisch-Westfälische Technische Hochschule Aachen

Tag der mündlichen Prüfung: 30.06.2008

Abstract

Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. The core step of the KDD process is the application of a *Data Mining* algorithm in order to produce a particular enumeration of patterns and relationships in large databases. *Clustering* is one of the major data mining techniques and aims at grouping the data objects into meaningful classes (clusters) such that the similarity of objects within clusters is maximized, and the similarity of objects from different clusters is minimized. This can serve to group customers with similar interests, or to group genes with related functionalities.

Currently, a challenge for clustering-techniques are especially high dimensional feature-spaces. Due to modern facilities of data collection, real data sets usually contain many features. These features are often noisy or exhibit correlations among each other. However, since these effects in different parts of the data set are differently relevant, irrelevant features cannot be discarded in advance. The selection of relevant features must therefore be integrated into the data mining technique.

Since about 10 years, specialized clustering approaches have been developed to cope with problems in high dimensional data better than classic clustering approaches. Often, however, the different problems of very different nature are not distinguished from one another. A main objective of this thesis is therefore a systematic classification of the diverse approaches developed in recent years according to their task definition, their basic strategy, and their algorithmic approach. We discern as main categories the search for clusters (i) w.r.t. closeness of objects in axis-parallel subspaces, (ii) w.r.t. common behavior (patterns) of objects in axis-parallel subspaces, and (iii) w.r.t. closeness of objects in arbitrarily oriented subspaces (so called correlation cluster).

For the third category, the remaining parts of the thesis describe novel approaches. A first approach is the adaptation of density-based clustering to the problem of correlation clustering. The starting point here is the first density-based approach in this field, the algorithm 4C. Subsequently, enhancements and variations of this approach are discussed allowing for a more robust, more efficient, or more effective behavior or even find hierarchies of correlation clusters and the corresponding subspaces. The density-based approach to correlation clustering, however, is fundamentally unable to solve some issues since an analysis of local neighborhoods is required. This is a problem in high dimensional data. Therefore, a novel method is proposed tackling the correlation clustering problem in a global approach. Finally, a method is proposed to derive models for correlation clusters to allow for an interpretation of the clusters and facilitate more thorough analysis in the corresponding domain science. Finally, possible applications of these models are proposed and discussed.

Zusammenfassung

Knowledge Discovery in Databases (KDD) ist der Prozess der automatischen Extraktion von Wissen aus großen Datenmengen, das gültig, bisher unbekannt und potentiell nützlich für eine gegebene Anwendung ist. Der zentrale Schritt des KDD-Prozesses ist das Anwenden von Data Mining-Techniken, um nützliche Beziehungen und Zusammenhänge in einer aufbereiteten Datenmenge aufzudecken. Eine der wichtigsten Techniken des Data Mining ist die Cluster-Analyse (Clustering). Dabei sollen die Objekte einer Datenbank in Gruppen (Cluster) partitioniert werden, so dass Objekte eines Clusters möglichst ähnlich und Objekte verschiedener Cluster möglichst unähnlich zu einander sind. Hier können beispielsweise Gruppen von Kunden identifiziert werden, die ähnliche Interessen haben, oder Gruppen von Genen, die ähnliche Funktionalitäten besitzen.

Eine aktuelle Herausforderung für Clustering-Verfahren stellen hochdimensionale Feature-Räume dar. Reale Datensätze beinhalten dank moderner Verfahren zur Datenerhebung häufig sehr viele Merkmale (Features). Teile dieser Merkmale unterliegen oft Rauschen oder Abhängigkeiten und können meist nicht im Vorfeld ausgesiebt werden, da diese Effekte in Teilen der Datenbank jeweils unterschiedlich ausgeprägt sind. Daher muss die Wahl der Features mit dem Data-Mining-Verfahren verknüpft werden.

Seit etwa 10 Jahren werden vermehrt spezialisierte Clustering-Verfahren entwickelt, die mit den in hochdimensionalen Feature-Räumen auftretenden Problemen besser umgehen können als klassische Clustering-Verfahren. Hierbei wird aber oftmals nicht zwischen den ihrer Natur nach im Einzelnen sehr unterschiedlichen Problemen unterschieden. Ein Hauptanliegen der Dissertation ist daher eine systematische Einordnung der in den letzten Jahren entwickelten sehr diversen Ansätze nach den Gesichtspunkten ihrer jeweiligen Problemauffassung, ihrer grundlegenden Lösungsstrategie und ihrer algorithmischen Vorgehensweise. Als Hauptkategorien unterscheiden wir hierbei die Suche nach Clustern (1.) hinsichtlich der Nähe von Cluster-Objekten in achsenparallelen Unterräumen, (2.) hinsichtlich gemeinsamer Verhaltensweisen (Mustern) von Cluster-Objekten in achsenparallelen Unterräumen und (3.) hinsichtlich der Nähe von Cluster-Objekten in beliebig orientierten Unterräumen (sogenannte Korrelations-Cluster).

Für die dritte Kategorie sollen in den weiteren Teilen der Dissertation innovative Lösungsansätze entwickelt werden. Ein erster Lösungsansatz basiert auf einer Erweiterung des dichte-basierten Clustering auf die Problemstellung des Korrelations-Clustering. Den Ausgangspunkt bildet der erste dichtebasierte Ansatz in diesem Bereich, der Algorithmus 4C. Anschließend werden Erweiterungen und Variationen dieses Ansatzes diskutiert, die robusteres, effizienteres oder effektiveres Verhalten aufweisen oder sogar Hierarchien von Korrelations-Clustern und den entsprechenden Unterräumen finden. Die dichtebasierten Korrelations-Cluster-Verfahren können allerdings einige Probleme grundsätzlich nicht lösen, da sie auf der Analyse lokaler Nachbarschaften beruhen. Dies ist in hochdimensionalen Feature-Räumen problematisch. Daher wird eine weitere Neuentwicklung vorgestellt, die das Korrelations-Cluster-Problem mit einer globalen Methode angeht. Schließlich wird eine Methode vorgestellt, die Cluster-Modelle für Korrelationscluster ableitet, so dass die gefundenen Cluster interpretiert werden können und tiefergehende Untersuchungen in der jeweiligen Fachdisziplin zielgerichtet möglich sind. Mögliche Anwendungen dieser Modelle werden abschließend vorgestellt und untersucht.

Contents

Part I

Preliminaries

Chapter 1

Introduction

Nowadays, the amount of data being collected in databases 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. Section 1.2 presents a detailed 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 [51]. The KDD process, as illustrated in Figure 1.1, consist of an iterative sequence of the following steps:

- **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 remov-

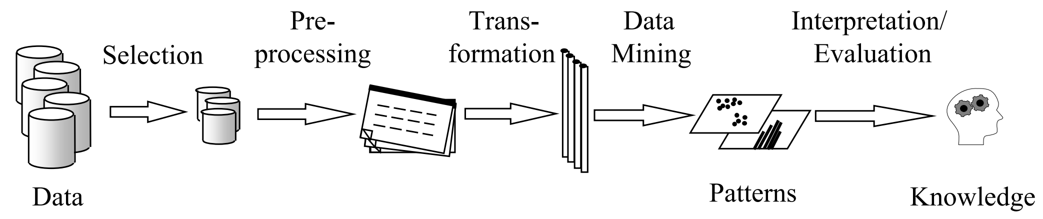


Figure 1.1: The KDD process.

ing 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 terms "KDD" and "Data Mining" are often used as synonyms. In [51], 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 [60]:

- **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.

In this thesis, we mainly focus on clustering techniques w.r.t. the special challenges posed by high dimensional data. In general, clustering aims at dividing data sets into subsets (clusters). Cluster analysis has been used in a large variety of fields such as astronomy, physics, medicine, biology, archaeology, geology, geography, psychology, and marketing. Many different research areas contributed new approaches (namely pattern recognition, statistics, information retrieval, machine learning, bioinformatics, and data mining). In some cases, the goal of cluster analysis is a better understanding of the data (e.g. learning the “natural” structure of data which should be reflected by a meaningful clustering). In other cases, cluster analysis is merely a first step for different purposes such as indexing or data compression.

While clustering in general is a rather dignified problem, mainly in about the last decade new approaches have been proposed to cope with new challenges provided by modern capabilities of automatic data generation and acquisition in more and more applications producing a vast amount of high dimensional data. These data need to be analyzed by data mining methods in order to gain the full potentials from the gathered information. However, high dimensional data pose different challenges for clustering algorithms that require specialized solutions. So, this area of research has been a highly active one in the recent years with a plethora of proposed algorithms but, in our opinion, lacking of a systematic problem analysis. Thus, a comparison of proposed algorithms is difficult both, theoretically and practically.

In this thesis, we aim at characterizing the different problems encountered when clustering high dimensional data. Afterwards, we describe our specialized solutions for some of these problems. The detailed outline is given in the following.

1.2 Outline

The content of this thesis is organized as follows:

Part I gives a general overview and introduction. The Introduction (Chapter 1) presents an overview on the filed of Knowledge Discovery in Databases and Data Mining in a very general manner to present the reader with the broader context of this thesis. The specialized field of clustering in high dimensional data is encountered in many fields of possible applications. Some prominent examples are sketched in Chapter 2 in order to first give the reader an impression of possible application scenarios. The remainder of the thesis presents the fundamental problems and corresponding solutions in a mere theoretical point of view, albeit proposed algorithms are always evaluated using synthetic and real world data.

Part II provides a deeper analysis of the problems encountered when clustering high dimensional data. The fundamental problem is sketched in Chapter 3 and characterizes shortly three different classes of clustering approaches in terms of their objectives. We discern as main categories the search for clusters (i) w.r.t. closeness of objects in axis-parallel subspaces, (ii) w.r.t. common behavior (patterns) of objects in axis-parallel subspaces, and (iii) w.r.t. closeness of objects in arbitrarily oriented subspaces (so called correlation cluster). Algorithms for these categories are surveyed and analyzed theoretically in Chapters 4, 5, and 6, respectively. Afterwards, the encountered problems and corresponding solutions are discussed in more detail in Chapter 7.

Part III is a collection of contributions of the author to the field of correlation clustering as enhancements of the density-based clustering paradigm. Chapter 8 describes the starting point for all these adaptations, the algorithm 4C. Further enhancements are described in the subsequent chapters for flat (Chapter 9) or hierarchical (Chapters 10 and 11) correlation clustering. Some weak points common to all PCA-based correlation clustering algorithms are addressed in Chapter 12 along with a framework as a remedy to the weaknesses.

However, there are inherent drawbacks of the density-based approach to correlation clustering. **Part IV** is dedicated to address these drawbacks. Chapter 13 surveys PCA-based approaches w.r.t. these common drawbacks. A novel approach to correlation clustering not suffering from the same problems is described in Chapter 14.

Part V moves on to the next step of the KDD-process following the data mining step: interpretation of the results. As discussed in Chapter 15, this step is not readily available for correlation clustering so far. Therefore, in Chapter 16, a model is proposed that can be used to interpret correlation clusters and to support the domain scientist in designing new and refined experiments. Possible other applications in data mining based on such models for correlation clusters are finally proposed in Chapter 17 (Classification) and Chapter 18 (Outlier Detection).

Part VI concludes the thesis summarizing the contributions and results (Chapter 19) and pointing out possible future directions (Chapter 20).

Chapter 2

Sample Applications for Clustering in High Dimensional Data

In many applications cluster analysis of high dimensional data is very important. Here, four probably rather prominent examples are described.

2.1 Gene Expression Analysis

Microarray chip technology produces a large amount of data in molecular biology. Microarray data — also called gene expression data — contain the "expression level" of genes measured under different conditions, e.g. in different tissues, under varying experimental environments, at different time slots after special treatment, or from different test persons. The expression level of a gene allows to draw conclusions about the amount of the corresponding gene product, such as a protein or a regulatory RNA, in the particular cell. Microarray data usually comprise the simultaneous measurement of the expression level of thousands of genes under hundreds of conditions. It consists of a data matrix where the rows correspond to genes and the columns represent different experimental conditions, different tissues, consecutive time slots, or different test persons. Biologists usually want to find patterns in such massive data sets. Depending on the scope of their research, the data mining task can vary.

Clustering the Rows. Very often, the biologists want to find groups of genes with homogeneous expression levels indicating that these genes share a common function. In that case, the columns usually represent different experimental conditions or different time slots within a time-dependent experiment. In general, genes may have very different functions depending on the cellular environment. Thus, the genes will usually cluster differently in varying subsets of the conditions or time slots. In other words, the clustering algorithm should take into account that a given gene A may e.g. be grouped together with gene B but not with gene C in a subset S of the columns, whereas A may e.g. be grouped together with C but not with B in another subset T of columns.

Clustering the Columns. In medical research, microarray data is often used to find genetic relationships and disorders. In that case, the columns of the gene expression data matrix represent different individuals. The data mining task is to cluster these individuals. Any clustering algorithm has to take into account, that the individuals usually differ in many phenotypical aspects, e.g. gender, age, hair color, specific diseases, etc. These different phenotypes are caused by different subsets of genes. In other words, e.g. the individual A may be grouped together with individual B but not with individual C in a subset S of genes, whereas A may be grouped together with C but not with B in another subset T of genes.

Co-clustering Rows and Columns. Especially in this application domain, the combination of both these tasks, simultaneously clustering the rows and the columns of a gene expression data matrix, is considered a specialized problem and a special family of algorithms is dedicated to solve it. However, the results of clustering rows and clustering columns can be translated into one another and both pose a classical problem description for subspace clustering.

2.2 Metabolic Screening

Many governments have implemented a metabolic screening of newborns in order to detect metabolic diseases in the earliest possible moment. For that purpose, a blood sample is taken from each newborn and the concentrations of specific metabolites, i.e. metabolic products, in these blood samples are measured. In the resulting data matrix, the rows represent the newborns and the columns represent the metabolites. Biologists usually want to identify homogeneous groups of newborns suffering from a common metabolic disease. Usually each metabolic disease causes a correlation between the concentration of a specific set of metabolites. Thus, any clustering algorithm should take into account that newborns should be grouped together only if they exhibit a common correlation among a set of metabolites. In addition, the set of participating metabolites and the type of correlation can be different for different diseases, i.e. clusters.

2.3 Customer Recommendation Systems

In customer recommendation systems, customers of a company can vote for the company's products. Depending on the portfolio of the company, there may be a very large set of products. It is now interesting, e.g. for target marketing purposes, to cluster the customers into groups of homogeneous voting schemata. Customers that have similar preferences should be grouped together. For each group, special marketing strategies can be applied taking each group's preferences into account. The problem for a cluster analysis process is that different customers may be grouped together according to different sets of products. In other words, customer A may share a preference for a given set S of products with customer B but not with C, whereas A may share another preference for a different set T of products with C but not with B. To make the problem even more challenging, the relationships between the preferences of the customers of one cluster may be arbitrary complex like “the lower the products p_1 and p_2 are rated, the higher the products p_3 and p_4 are rated”.

2.4 Text Documents

Clustering collections of text documents such as web pages to find groups of thematically related documents is important in many applications. Usually, the text documents are transformed into high dimensional feature vectors, e.g. using term frequency features. The data matrix to be analyzed then contains each document as a row where the columns represent the count of one particular term in the corresponding document. Since the count for any term occurring in any text document (after excluding stop words, applying stemming, etc.) needs to be recorded, usually the data contain thousands of attributes and is thus very sparse featuring a lot of zero values. Again, related documents will only have a similar word count in a subset of terms, and these subsets are likely to be different for different groups of thematically relevant documents. In addition, the thematic groups may overlap, i.e. one document may be assigned to more than one thematic group according to similarities of the count value in different subsets of terms. In other words, document A may share a similar frequency in a given set S of terms with document B but not with C, whereas A may share another similar frequency in a different set T of terms with C but not with B.

Part II

Typical Problems and Solutions in Clustering High Dimensional Data

Since almost all research on clustering high dimensional data is relatively new, it is not covered in most textbooks in different related fields (data mining, statistics, machine learning, pattern recognition – cf. e.g. [61, 67, 141, 32]) or in not-so-recent surveys on the topic of clustering [74]. Others [46, 59, 130] sketch the problem rather casually. A more recent edition [60] at least dedicates a section to the problem sketching some example algorithms and touching on some problems.

Recently, some surveys already have given overviews on some approaches. In [109], the problem is introduced in a very illustrative way and some approaches are sketched. However, there is no clear distinction between different subproblems (axis-parallel or arbitrarily oriented) and the corresponding algorithms are discussed without pointing out the underlying differences in the corresponding problem definitions. Van Mechelen et al. [137] gave an overview on older, more specialized work of a special type of pattern-based clustering (biclustering) in the medical and biological area from a statistics point of view. Madeira and Oliveira [97] focus on pattern-based clustering approaches and are especially interested in the application domain of microarray data. Jiang et al. [75] focus exclusively on the application domain of gene expression data and discuss clustering approaches structured according to the application scenarios. However, in addition to the applications of full dimensional clustering approaches, they sketch only three biclustering approaches (named "subspace clustering" in their overview). Tanay et al. [132] discuss some biclustering algorithms as representatives of different algorithmic approaches, also focussed on the application to gene expression data.

Here, we would like to give a more systematic approach to the problem and on the different tasks and subproblems (axis-parallel, pattern-based, arbitrarily oriented clustering). Therefore, we will also survey the related heuristics used by different approaches. Our systematic view is not based on the application scenarios but on the intrinsic methodological differences of the various families of approaches based on different spatial intuitions. Thus, we will also try to integrate the inherently different point of view of pattern-based approaches into the intuition of patterns in data space.

In this part, we will first grasp the problems in clustering high dimensional data in a general way (Chapter 3). Concluding this introductory survey, we distinguish three basic classes of clustering algorithms for high dimensional data. These categories will be surveyed in detail in the subsequent chapters (Chapter 4–6).

Having introduced the different concepts and approaches, we will discuss the approaches again in a more general way comparing the different problem-statements as well as the heuristics and the related restrictions of the different approaches (cf. Chapter 7).

The systematic overview given in this part has been presented as a tutorial at ICDM 2007 [89].

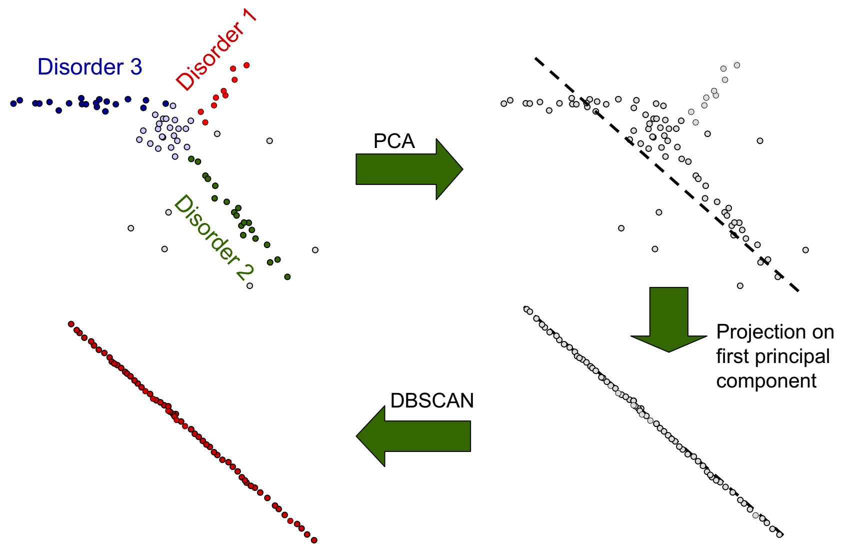
Chapter 3

Finding Clusters in High Dimensional Data

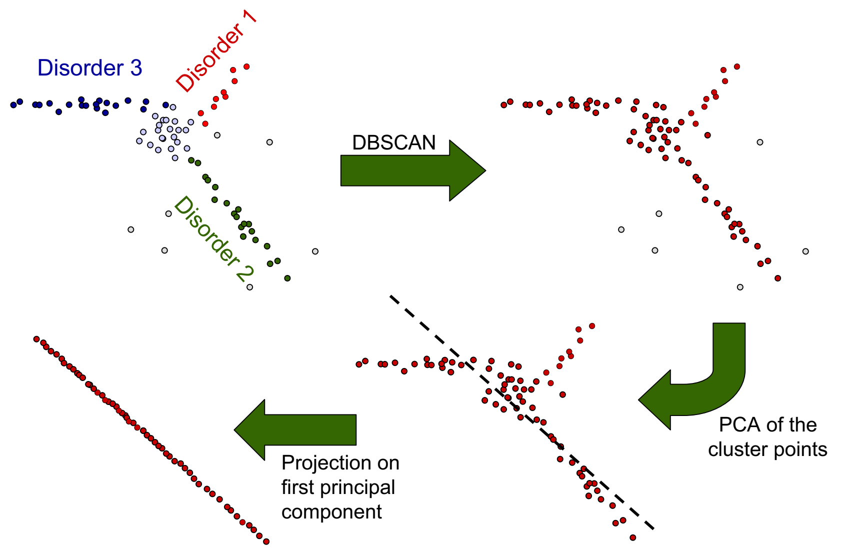
When clustering high dimensional data, we face different problems. The presence of irrelevant features or of correlations among subsets of features heavily influences the appearance of clusters in the full dimensional space. The main challenge for clustering here is that different subsets of features are relevant for different clusters, i.e. the objects cluster in subspaces of the data space but the subspaces of the clusters may vary. Additionally, different correlations among the attributes may be relevant for different clusters. We call this phenomenon that different features or a different correlation of features may be relevant for varying clusters *local feature relevance* or *local feature correlation*.

A common way to overcome problems of high dimensional data spaces where several features are correlated or only some features are relevant is to perform feature selection before performing any other data mining task. Feature selection methods like principal component analysis (PCA) can be used to map the original data space to a lower dimensional data space where the points may cluster better and the resulting clusters may be more meaningful.

Unfortunately, such feature selection or dimensionality reduction techniques cannot be applied to clustering problems. Feature selection or dimen-



(a) First dimensionality reduction then clustering.


Disorder 3
Disorder 1
Disorder 2
DBSCAN
PCA of the cluster points
Projection on first principal component

(b) First clustering then dimensionality reduction.

Figure 3.1: Illustration of the local feature relevance/local feature correlation problem.

sionality reduction techniques are *global* in the following sense: they generally compute only one subspace of the original data space in which the clustering can then be performed. In contrast, the problem of *local feature relevance* and *local feature correlation* states that multiple subspaces are needed because each cluster may exist in a different subspace. Figure 3.1(a) illustrates this problem for a fictive sample two-dimensional data set derived from the metabolic screening application. The data contain a set of patients, some of them healthy, others suffering from specific metabolic diseases. For each patient, the concentrations of two fictive metabolites are measured. There are four clusters (healthy, disorder 1 – 3) and some noise. Whereas the cluster of the healthy patients form a conventional two-dimensional cluster, for each cluster representing ill patients different feature relevance and feature correlation applies: for disorder 1, a positive correlation between both attributes can be observed. For disorder 2, a negative correlation between both attributes can be observed. For disorder 3, only the feature represented by the y-axis is relevant. If a feature selection method (here PCA) is applied to these data in order to reduce the dimensionality by one, the four clusters cannot be separated in the resulting subspace anymore (cf. Figure 3.1(a)). On the other hand, if the points in the original data space are clustered first (here using DBSCAN [47]) and afterwards feature selection is applied to each resulting cluster (e.g. again using PCA), also no reasonable result can be detected (cf. Figure 3.1(b)). In summary, due to the problem of local feature relevance and local feature correlation, usually no global feature selection can be applied to overcome the challenges of clustering high dimensional data.

Instead of a global approach to feature selection, a local approach accounting for the local feature relevance and/or local feature correlation problems is required. Since traditional methods like feature selection, dimensionality reduction, and conventional clustering do obviously not solve the above sketched problems, novel methods need to integrate feature analysis into the clustering process more tightly. Figure 3.2 illustrates the general challenge for finding clusters in high dimensional data. Cluster 3 exists in an axis-parallel subspace, clusters 1 and 2 exist in (different) arbitrarily oriented subspaces: if the cluster members are projected onto the depicted subspaces, the points

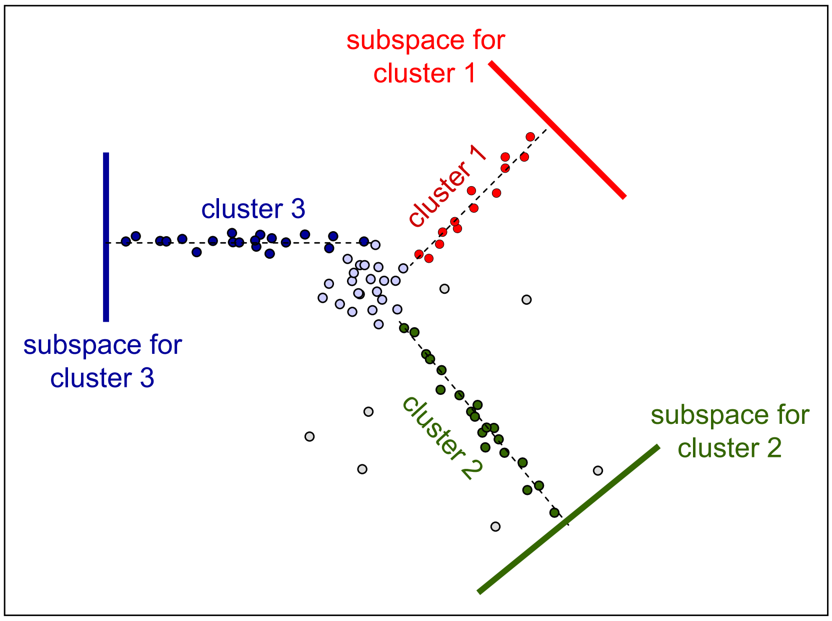


Figure 3.2: Illustration of the general aim of clustering algorithms for high dimensional data.

are densely packed, i.e. similar to each other. Generally, we can derive the following aim for methods that are designed for clustering high dimensional data:

> *The general aim of clustering algorithms designed for high dimensional data is to find clusters in arbitrarily oriented subspaces of the original feature space.*

Of course, the meaning of the term “clusters” is still open to debate. Since there are different approaches to define a “cluster” in general, there are also different notions of what constitutes a “subspace cluster”. But assuming a certain meaning of “cluster”, we discuss different general algorithmic approaches to finding clusters in high dimensional data.

A naïve solution to the general aim of clustering algorithms for high dimensional data is to test all possible arbitrarily oriented subspaces for clusters. Obviously, there is an infinite number of arbitrarily oriented subspaces, so this naïve solution is computationally infeasible. Rather, we urgently need some heuristics and assumptions in order to conquer this infinite search space.

As Figure 3.2 suggests, the clusters can in general be found in arbitrarily oriented subspaces. However, in some applications, it is reasonable to focus only on clusters in axis-parallel subspaces (like cluster 3 in Figure 3.2). In that case, the search space of all potential subspaces accommodating clusters is restricted, but still in $O(2^d)$. Many algorithms proposed so far use this restriction and are limited to finding clusters in axis-parallel subspaces only. In the literature, those clustering algorithms are usually called *projected clustering* or *subspace clustering* algorithms. We review and discuss the problem of finding clusters in axis-parallel subspaces in Chapter 4.

On the other hand, several applications require solutions for the general case where clusters may exists in any arbitrarily oriented subspace. The algorithms of this class of solutions are usually called correlation clustering algorithms. Let us note that some authors use the term "subspace clustering algorithm" interchangeably also for "correlation clustering algorithm". We review and discuss the problem of finding clusters in arbitrarily oriented subspaces in Chapter 6.

In between these two main classes of existing algorithms, finding clusters in axis-parallel subspaces and clusters in arbitrarily oriented subspaces, a third class of algorithms following a slightly different approach has been proposed. Those algorithms are typically referred to as pattern-based clustering (or sometimes: bi-clustering, co-clustering) algorithms. In fact, some pattern-based clustering algorithms are restricted to axis-parallel subspace clusters, whereas other pattern-based clustering algorithms are not restricted to axis-parallel subspace clusters but are limited to clusters in special cases of arbitrarily oriented subspaces. However, since the pattern-based clustering methods take a different approach to the problems presented here and expose a kind of hybrid approach between axis-parallel and arbitrarily oriented subspace clustering, we discuss them as a separate class of algorithms in Chapter 5.

Chapter 4

Finding Clusters in Axis-parallel Subspaces

A very common assumption to shrink down the infinite search space of all possible subspaces is to focus on axis-parallel subspaces only. The assumption that clusters can only be found in axis-parallel subspaces may be rather sensible in the context of various applications. The big advantage is that the search space is now restricted by the number of all possible axis-parallel subspaces. However, the bound is still rather high: In a d-dimensional data set the number of k-dimensional subspaces is $\binom{d}{k}$ $(1 \leq k \leq d)$ and, thus, the number of all possible subspaces is

$$\sum_{k=1}^{d} \binom{d}{k} = 2^d - 1.$$

In the literature, the problem of finding axis-parallel clusters has been referred to as "projected clustering" and "subspace clustering". However, these terms are not consistently used in the literature causing some potential misunderstanding. Here, we try to establish a standard vocabulary. Originally, projected clustering and subspace clustering refer to two different sub-problems of finding clusters in axis-parallel subspaces (or projections). However, in the literature, these two sub-problems have been mixed up e.g. with the algorithmic approach used to conquer the search space of possible subspaces to look for clusters. For example, the distinction between "dimension-growth subspace clustering" and "dimension-reduction projected clustering" given in [60] is such a mix-up. In the following, we give two classification schemata of existing algorithms to find clusters in axis-parallel subspaces. The first schema is a problem-oriented view of the task and ends up in defining the terms "projected clustering" and "subspace clustering" in a unified way. The second schema is an algorithmic view analyzing the algorithmic approach employed to conquer the exponential search space of possibly interesting subspaces.

4.1 A Problem-Oriented Categorization

Due to the afore mentioned exponential search space, all algorithms that are limited to finding clusters in axis-parallel subspaces rely on further assumptions that usually affect the results produced. In the literature, there are generally three different classes of problem statements depending on the assumptions made.[1]

1. **Projected Clustering Algorithms**
   A first class of algorithms aims at finding a unique assignment of each point to exactly one subspace cluster (or noise). Generally, they try to find the projection where the currently considered set of points clusters best. These algorithms are referred to as *projected clustering* algorithms. Some algorithms further assume that the number k of clusters is known beforehand such that an objective function can be defined which is optimized to derive the optimal set of k clusters.

2. **Subspace Clustering Algorithms**
   A second class of algorithms aims at finding all subspaces where clusters can be identified. Thus, these algorithms are dedicated to find all clusters in all subspaces. We refer to this group of algorithms as *subspace clustering* algorithms.

[1]Note that problem statements are often not stated explicitly.

3. **Hybrid Algorithms**
   A third class of algorithms aims at finding something in between. Usually, these algorithms aim at finding clusters that may overlap. On the other hand, these algorithms do not aim at finding *all* clusters in *all* subspaces. Some of the hybrid algorithms only compute interesting subspaces rather than final subspace clusters. The reported subspaces can then be mined by applying full dimensional algorithms to these projections.

Let us note that all classes of algorithms imply that there is a definition of what constitutes a cluster. The output of these algorithms is a list of clusters each represented as a pair (X, Y), where X is a subset of data objects and Y is a subset of data attributes, such that the points in X meet a given cluster criterion when projected onto the attributes in Y but do not meet the cluster criterion when projected onto the remaining attributes, i.e. the points in X are "close" when projected onto the attributes in Y but projected onto the remaining attributes they are "not close". Usually, the cluster criterion and the measure of "closeness" differs from algorithm to algorithm.

4.2 An Algorithmic-Oriented Categorization

A second classification schema of existing algorithms for finding clusters in axis-parallel subspaces focuses on the algorithmic approach to conquer the exponential search space of all possible subspaces. In general, this is an important view because efficiently navigating through this search space is one of the key challenges for the design of an axis-parallel projected, subspace or hybrid clustering algorithm. The task is to efficiently identify those subspaces, that accommodate one or more clusters. Compared to evaluating a given cluster criterion, the search for the subspaces accommodating a cluster is usually the bottleneck even for low dimensional data. For example, for $d = 20$, we face more than 1 million possible subspaces. A complete enumeration of these subspaces is obviously computationally infeasible.

In general, the algorithmic approaches for finding these subspaces, i.e.

traversing the search space of all possible axis-parallel subspaces, can be divided into the following two categories.

1. **Top-down Approaches**
   The rational of top-down approaches is to determine the subspace of a cluster starting from the full dimensional space. This is usually done by determining a subset of attributes for a given set of points – potential cluster members – such that the points meet the given cluster criterion when projected onto the corresponding subspace. Obviously, the dilemma is, that for the determination of the subspace of a cluster, at least some cluster members must be identified. On the other hand, in order to determine cluster memberships, the subspace of each cluster must be known. To escape from this circular dependency, most of the top-down approaches rely on a rather strict assumption, which we call the *locality assumption*. It is assumed that the subspace of a cluster can be derived from the local neighborhood (in the full dimensional data space) of the cluster center or the cluster members. In other words, it is assumed that even in the full dimensional space, the subspace of each cluster can be learned from the local neighborhood of cluster representatives or cluster members. Other top-down approaches that do not rely on the locality assumption use random sampling in order to generate a set of potential cluster members.

2. **Bottom-up Approaches**
   The exponential search space that needs to be traversed is equivalent to the search space of the frequent item set problem in market basket analysis in transaction databases [14]. Each attribute represents an item and each subspace cluster is a transaction of the items representing the attributes that span the corresponding subspace. Finding itemsets with frequency 1 then relates to finding all combinations of attributes that constitute a subspace containing at least one cluster. This observation is the rational of most bottom-up subspace clustering approaches. The subspaces that contain clusters are determined starting from all 1-dimensional subspaces that accommodate at least

one cluster by employing a search strategy similar to frequent itemset mining algorithms. To apply any efficient frequent itemset mining algorithm, the cluster criterion must implement a downward closure property (also called monotonicity property): *If subspace S contains a cluster, then any subspace $T \subseteq S$ must also contain a cluster.* The reverse implication, *if a subspace T does not contain a cluster, then any superspace $S \supseteq T$ also cannot contain a cluster*, can be used for pruning, i.e. excluding specific subspaces from consideration. Let us note that there are bottom-up algorithms that do not use an APRIORI-like subspace search but instead apply other search heuristics.

Both the top-down approach and the bottom-up approach are commonly used in the literature. While the top-down approach tries to anticipate cluster members and then determines the subspace of each cluster, the bottom-up approach rather tries to anticipate the subspaces of the clusters and then determines the cluster members.

4.3 Survey and Categorization of Existing Approaches

In the following, we survey representative solutions, categorized according to the task-definition they adopt.

4.3.1 Projected Clustering Algorithms

Projected clustering algorithms aim at finding a unique assignment of points to subspace clusters. Some algorithms also model noise explicitly, i.e. points are assigned uniquely to only one cluster or the noise set.

PROCLUS [10] is a k-medoid-like clustering algorithm. It randomly determines a set of potential cluster centers M on a sample of points first. In the iterative cluster refinement phase, for each of the k current medoids the subspace is determined by minimizing the standard deviation of the distances of the points in the neighborhood of the medoids to the corresponding medoid along each dimension. Points are then assigned to the closest medoid considering the relevant subspace of each medoid. The clusters are refined by replacing bad medoids with new medoids from M as long as the clustering quality increases. A postprocessing step identifies noise points that are too far away from their closest medoids. The algorithm always outputs a partition of the data points into k clusters (each represented by its medoid) with corresponding subspaces and a (potentially empty) set of noise points. The k-medoid-style cluster model tends to produce equally sized clusters that have spherical shape in their corresponding subspaces. In addition, since the set M of possible medoids is determined in a randomized procedure, different runs of PROCLUS with the same parametrization usually result in different clusterings. A similar method, LAC (Locally Adaptive Clustering) [44], starts with k centroids and k sets of d weights (for d attributes). The algorithm proceeds to approximate a set of k Gaussians by adapting the weights. The difference to PROCLUS is that weights are computed for all attributes while for PROCLUS the average cluster dimensionality needs to be specified. Other variations of PROCLUS are FINDIT [142] employing additional heuristics to enhance efficiency and clustering accuracy and SSPC [146] that offers the capability of further enhancing accuracy by using domain knowledge in the form of labeled objects and/or labeled attributes.

PreDeCon [34] applies the density-based full dimensional clustering algorithm DBSCAN [47] using a specialized distance measure that captures the subspace of each cluster. The definition of this specialized subspace distance is based on the so-called subspace preference that is assigned to each point $\vec{p}$, representing the maximal-dimensional subspace in which $\vec{p}$ clusters best. A dimension is considered to be relevant for the subspace preference of a point $\vec{p}$ if the variance of points in the Euclidean ε-neighborhood of $\vec{p}$ is below a user-defined threshold δ. The specialized subspace distance between points is a weighted Euclidean distance where the dimensions relevant for the subspace preference of a point are weighted by a constant $\kappa \gg 1$ while the remaining dimensions are weighted by 1. PreDeCon determines the number of clusters automatically, and handles noise implicitly. In addition, its results are determinate and the clusters may exhibit any shape and size in the corresponding subspace. However, PreDeCon requires the user to specify a number of input parameters that are usually hard to guess.

CLTree [94] is a method that presents an interesting variation of the theme. The basic idea is to assign a common class label to all existing points and to add additionally points uniformly distributed over the data space and labeled as different class. Then a decision tree is trained to separate the two classes. As a consequence, the attributes are split independently, adaptively, and in a flexible order of the attributes. However, selecting a split is based on the evaluation of information gain which is rather costly. Furthermore, the density of the superimposed artificial data can be expected to heavily influence the quality of the results. Since the distribution parameters of existing clusters are unknown beforehand, finding a suitable parametrization seems rather hard. Another problem is the merging of adjacent regions. A cluster can easily become separated if the corresponding bins do not "touch" each other.

4.3.2 Subspace Clustering Algorithms

Subspace clustering algorithms aim at finding all clusters in all subspaces of the entire feature space.

CLIQUE [13], the pioneering approach to subspace clustering, uses a grid-based clustering notion. The data space is partitioned by an axis-parallel grid into equi-sized units of width ξ. Only units which contain at least τ points are considered as dense. A cluster is defined as a maximal set of adjacent dense units. Since dense units satisfy the downward closure property, subspace clusters can be explored rather efficiently in a bottom-up way. Starting with all 1-dimensional dense units, $(k + 1)$-dimensional dense units are computed from the set of k-dimensional dense units in an APRIORI-like style. If a $(k + 1)$-dimensional unit contains a projection onto a k-dimensional unit that is not dense, then the $(k + 1)$-dimensional unit can also not be dense. Furthermore, a heuristic that is based on the minimum description length principle is introduced to discard candidate units within less interesting subspaces, i.e. subspaces that contain only a very small number of dense units. This way, the efficiency of the algorithm is enhanced but at the cost of incomplete results, i.e. some true clusters are lost. There are some variants of CLIQUE. The method ENCLUS [40] also relies on a fixed grid but searches for subspaces that potentially contain one or more clusters rather than for dense units. Three quality criteria for subspaces are introduced, one of them implements the downward closure property. The method MAFIA [105] uses an adaptive grid. The generation of subspace clusters is similar to CLIQUE. Another variant of CLIQUE called nCluster [95] allows overlapping windows of length δ as 1-dimensional units of the grid. In summary, all grid-based methods use a simple but rather efficient cluster model. The shape of each resulting cluster corresponds to a polygon with axis-parallel lines in the corresponding subspace. Obviously, the accuracy and the efficiency of CLIQUE and its variants primarily depend on the granularity and the positioning of the grid. A higher grid granularity results in higher runtime-requirements but will most likely produce more accurate results.

SUBCLU [80] uses the DBSCAN cluster model of density-connected sets [47]. It is shown that density-connected sets satisfy the downward closure property. This enables SUBCLU to search for density-based clusters in subspaces in an APRIORI-like style. The resulting clusters may exhibit an arbitrary shape and size in the corresponding subspaces. In fact, for each subspace SUBCLU computes all clusters that would have been found by DBSCAN applied to that subspace only. Compared to the grid-based approaches, SUBCLU achieves a better clustering quality but requires a higher runtime.

It has been observed that a global density threshold, as used by SUBCLU and the grid-based approaches, leads to a bias towards a certain dimensionality: a tighter threshold which is able to separate clusters from noise well in low dimensions tends to loose clusters in higher dimensions whereas a more relaxed threshold which is able to detect high dimensional clusters will produce an excessive amount of low dimensional clusters. Therefore, the dimensionality unbiased cluster model DUSC has been proposed, based on a density measure adaptive to the dimensionality [18]. As a major drawback, this approach is lacking of anti-monotonic properties and, thus, pruning the search space is not possible. A "weak density" is thus defined as a remedy, providing anti-monotonic properties. This remedy, however, in turn introduces a global density threshold again. A method for visual subspace cluster analysis based on DUSC is proposed in [19].

4.3.3 Hybrid Clustering Algorithms

Algorithms that do not aim at uniquely assigning each data point to a cluster and do not aim at finding all clusters in all subspaces are called hybrid algorithms. Some hybrid algorithms offer the user an optional functionality of a pure projected clustering algorithm. Others aim at computing only the subspaces of potential interest rather than the final clusters. Usually, hybrid methods that report clusters allow overlapping clusters but do not aim at computing all clusters in all subspaces.

DOC [114] uses a global density threshold to define a subspace cluster by means of hypercubes of fixed side-length w containing at least α points. A random search algorithm is proposed to compute such subspace clusters from a starting seed of sampled points. A third parameter β specifies the balance between the number of points and the dimensionality of a cluster. This parameter affects the dimensionality of the resulting clusters and, thus, DOC usually has also problems with subspace clusters of significantly different dimensionality. Due to the very simple clustering model, the clusters may contain additional noise points (if w is too large) or not all points that naturally belong to the cluster (if w is too small). One run of DOC may (with a certain probability) find one subspace cluster. If k clusters need to be identified, DOC has to be applied at least k times. If the points assigned to the clusters found so far are excluded from subsequent runs, DOC can be considered as a pure projected clustering algorithm because each point is uniquely assigned to one cluster or to noise (if not assigned to a cluster). On the other hand, if the cluster points are not excluded from subsequent runs, the resulting clusters of multiple runs may overlap. Usually, DOC cannot produce all clusters in all subspaces.

MINECLUS [147, 148] is based on a similar idea as DOC, but proposes a deterministic method to find an optimal projected cluster given a sample seed point. The authors transform the problem into a frequent item set mining problem and employ a modified frequent pattern tree growth method. Further heuristics are introduced to enhance efficiency and accuracy.

COSA [53] does not derive a clustering but merely a similarity matrix that can be used by an arbitrary clustering algorithm afterwards. The matrix contains weights for each point specifying a subspace preference of the points similar to PreDeCon. The weights for a point $\vec{p}$ are determined by starting with the Euclidean k-nearest neighbors of $\vec{p}$ and by computing the average distance distribution of the k-nearest neighbors along each dimension. As long as the weight vectors still change, the k-nearest neighbors are again determined using the current weights and the weights are re-computed. The number of neighbors k is an input parameter. Very different to PreDeCon, the weights can have arbitrary values rather than only two fixed values. In addition, the authors in [53] test the weighting matrix using several full dimensional clustering algorithms rather than integrating it into only one specific algorithm.

DiSH [3] follows a similar idea as PreDeCon but uses a hierarchical clustering model. This way, hierarchies of subspace clusters can be discovered, i.e. the information that a lower dimensional cluster is embedded within a higher dimensional one. The distance between points and clusters is defined in such a way that it reflects the dimensionality of the subspace that is spanned by combining the corresponding subspace of each cluster. As in COSA, the weighting of attributes is learned for each object, not for entire clusters. The learning of weights, however, is based on single attributes, not on the entire feature space. DiSH uses an algorithm that is inspired by the density-based hierarchical clustering algorithm OPTICS [16]. However, DiSH extends the cluster ordering computed by OPTICS in order to find hierarchies of subspace clusters with multiple inclusions (a lower dimensional subspace cluster may be embedded in multiple higher dimensional subspace clusters).

HARP [145] is a Single-Link like hierarchical clustering algorithm but uses a different "distance function" between points/clusters and does not produce a hierarchy of subspace clusters. Starting with singleton clusters, HARP iteratively merges clusters as long as the resulting cluster has a minimum number of relevant attributes. A relevance score is introduced for attributes based on a threshold that starts at some harsh value and is progressively decreased while clusters increase in size. By design, HARP has problems to find low dimensional clusters. The resulting dendrogram can be cut at any level in order to produce a unique assignment of points to clusters.

SCHISM [124] mines interesting subspaces rather than subspace clusters, thus, it is not exactly a subspace clustering algorithm but solves a related problem: find subspaces to look for clusters. It employs a grid-like discretization of the database and applies a depth-first search with backtracking to find maximally interesting subspaces.

FIRES [88] computes 1-dimensional clusters using any clustering technique the user is most accomplished to in a first step. These 1-dimensional clusters are then merged by applying a "clustering of clusters". The similarity of clusters is defined by the number of intersecting points. The resulting clusters represent hyper-rectangular approximations of the true subspace clusters. In an optional postprocessing step, these approximations can be refined by again applying any clustering algorithm to the points included in the approximation projected onto the corresponding subspace. Though using a bottom-up search strategy, FIRES is rather efficient because it does not employ a worst-case exhaustive search procedure but a heuristic that is linear in the dimensionality of the data space. However, this performance boost is paid for by an expected loss of clustering accuracy. It cannot be specified whether the subspace clusters produced by FIRES may overlap or not. In general, the clusters may overlap but usually, FIRES cannot produce all clusters in all subspaces.

P3C [102, 103] starts with 1-dimensional intervals that are likely to approximate higher dimensional subspace clusters. These intervals are merged using an APRIORI-like bottom-up search strategy. The maximal dimensional subspace cluster approximations resulting from this merging procedure are reported as so-called cluster cores. In a refinement step, the cluster cores are refined by using an EM-like clustering procedure. Each cluster core is taken as one initial cluster for the EM algorithm. Points are assigned to the closest cluster core using the Mahalanobis distance. The final output of P3C is a matrix that records for each data point its probability of belonging to each projected cluster. From this matrix, a disjoint partitioning of the data points into clusters can be obtained by assigning each point to the cluster with the highest probability. If overlapping clusters shall be allowed, each point can be assigned to all clusters with a probability larger than $1/k$. P3C cannot produce all clusters in all subspaces.

4.4 Summary

Two schemata for the classification of algorithms for finding clusters in axis-parallel subspaces have been presented. The first schema classifies the approaches according to the definition of the problem the algorithms aim to solve into *projected clustering*, *subspace clustering* and *hybrid algorithms*. The second schema distinguishes the algorithmic method to find the subspaces that accommodate the clusters, *bottom-up* vs. *top-down* approaches. In fact, there is a close relationship between the problem-oriented classification and the algorithmic-oriented classification. Many projected clustering algorithms implement a top-down approach, whereas all subspace clustering algorithms follow a bottom-up approach. This close connection explains the additions “dimension-growth” and “dimension-reduction” in the distinction between “dimension-growth subspace clustering” and “dimension-reduction projected clustering” in [60]. However, this relationship does not hold in general. In addition, for hybrid approaches there is no close relationship to any of the algorithmic-oriented classes, i.e. some implement a bottom-up approach, others use a top-down strategy.

Table 4.1: Categorization of sample subspace clustering algorithms, projected clustering algorithms, hybrid approaches

		algorithmic-oriented view	
	category	**bottom-up**	**top-down**
problem-oriented view	**subspace clustering**	CLIQUE nCluster ENCLUS MAFIA SUBCLU	
	hybrid	DiSH FIRES P3C SCHISM	DOC MINECLUS COSA HARP
	projected clustering	P3C	PROCLUS SSPC PreDeCon DOC MINECLUS

A classification of existing approaches has been presented following the classification w.r.t. their assumed task definition. Table 4.1 overviews the different categorizations of algorithms and their relationships also for the algorithmic point of view. Note that DOC, MINECLUS, and P3C appear multiple times since they can optionally produce overlapping or non-overlapping clusters.

In general, focussing on axis-parallel subspaces is meaningful in several applications. Since all existing approaches are based on further assumptions, a fair and comprehensive experimental comparison of all approaches is a large challenge. However, to decide which algorithm should be chosen for which task, such a comparison is urgently needed. Leastwise, this survey is an attempt to provide such a comparison from the theoretical point of view, sketching the different assumptions and heuristics used by the various approaches.

Chapter 5

Finding Clusters Based on Patterns in the Data Matrix

Recall the cluster definition of subspace and projected clustering algorithms: A clustering can be described as a set of pairs (X, Y), where X is a subset of data objects and Y is a subset of data attributes, such that the points in X are "close" when projected onto the attributes in Y but projected onto the remaining attributes they are "not close". Since the measure of "closeness" is unspecified by the problem definition of subspace and projected clustering in principle, also most of the pattern-based clustering algorithms could be interpreted as subspace or projected clustering algorithms in the above sense. The clustering algorithms surveyed in Chapter 4 usually define the "closeness" in a sense of density in terms of the Euclidean distance in an axis-parallel projection. The pattern-based clustering algorithms, as we will see in this section, define the "closeness" differently in the sense of a common behavior of objects in an axis-parallel subspace, i.e., w.r.t. a certain "pattern" which the objects form in a subset of attributes.

Let us embark upon discussing pattern-based clustering algorithms with a general consideration. We have seen that the heuristics used in subspace and projected clustering treat dimensions and points differently. Why are those directions not interchangeable? A reason may be that heuristics for speed-up are based on different intuitions for data space and data objects.

Furthermore, dependent on the problem at hand, the spatial intuition may be natural. Thus, data space and data objects are indeed different concepts for many applications. However, it is a general characteristic of pattern-based clustering algorithms (thus also called biclustering, co-clustering, two-mode clustering, or two-way clustering algorithms) that they treat attributes and objects interchangeable.[1]

While we claim to provide a rather thorough overview of existing approaches in Chapters 4 and 6, in this Chapter we aim merely at pointing out the connections among different biclustering models and their relationships to the more general approaches based on spatial intuitions. To present an overview on different models and connect those models to spatial intuitions, we follow the structure presented by Madeira and Oliveira [97] and try to enrich the rather abstract notions of bicluster types by intuitions what the patterns in a data matrix mean in the original data space. This will lead us to the surprising perception that, in terms of general subspace clustering approaches, many approaches in this field tackle rather simple, very specialized or even weird problems. For a more exhaustive covering of biclustering algorithms we refer to the afore mentioned surveys covering biclustering in biological and medical applications [97, 137, 132]. Recent work on pattern-based clustering is especially popular in the bioinformatics community focussing on the applications of biclustering on microarray data, triggered by [41].

5.1 General Aim and Basic Approaches of Pattern-based Clustering Algorithms

Pattern-based clustering algorithms depict the data as a matrix $\boldsymbol{A}$ with a set of rows X and a set of columns Y. The element a_{xy} represents the value in row x and column y. Usually, the rows represent database objects, the columns

[1]Note that there are counter examples, though. The algorithm MaPle [110] (see below) enumerates attributes first, based on the reasoning that there are usually much more objects than attributes in a database.

are the attributes of the database objects. Thus, the matrix element a_{xy} is the value of object with ID x in the attribute with ID y. We can consider such a matrix $\boldsymbol{A}$, with n rows and m columns, defined by its set of rows, $X = \{x_1, \ldots, x_n\}$, and its set of columns, $Y = \{y_1, \ldots, y_m\}$. Thus, we can denote the matrix $\boldsymbol{A}$ by (X, Y). Choosing $I \subseteq X$ and $J \subseteq Y$ as subsets of the rows and columns, respectively, $\boldsymbol{A}_{IJ} = (I, J)$ denotes the submatrix of $\boldsymbol{A}$ containing those elements a_{ij} with $i \in I$ and $j \in J$. Biclustering algorithms tackle the problem of finding a set of submatrices $\{(I_1, J_1), \ldots, (I_k, J_k)\}$ of the matrix $\boldsymbol{A} = (X, Y)$ (with $I_i \subseteq X, J_i \subseteq Y \ \forall i \in \{1, \ldots, k\}$), where each submatrix (bicluster) meets a given homogeneity criterion.

Many approaches make use of mean values of rows, of columns, and of the complete data matrix or a certain submatrix (i.e., a bicluster). For these mean values, the following notations are commonly in use. The mean of the ith row in the bicluster (I, J) is given by

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}. \tag{5.1}$$

The mean of the jth column in the bicluster (I, J) is given by

$$a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}. \tag{5.2}$$

The mean of all elements in the bicluster (I, J) is given by

$$\begin{aligned} a_{IJ} &= \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} && (5.3) \\ &= \frac{1}{|I|} \sum_{i \in I} a_{iJ} && (5.4) \\ &= \frac{1}{|J|} \sum_{j \in J} a_{Ij}. && (5.5) \end{aligned}$$

Madeira and Oliveira [97] discern basically four different categories of biclusters, *constant biclusters*, *biclusters with constant values on either columns or rows*, *biclusters with coherent values*, and finally *biclusters with coherent evolutions*. The general problem settings for these categories is discussed in the following.

5.1.1 Constant Biclusters

A *perfect* constant bicluster consists of points sharing *identical* values in all selected attributes. In the corresponding submatrix (I, J) it holds therefore for a constant value μ which is typical for the cluster and for all $i \in I$ and $j \in J$:

$$a_{ij} = \mu. \tag{5.6}$$

In a *not-so-perfect* constant bicluster the values are only similar but not necessarily identical, i.e.:

$$a_{ij} \approx \mu. \tag{5.7}$$

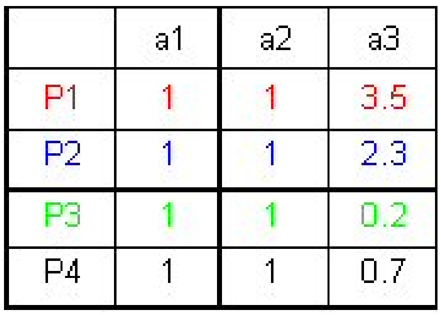
This type of bicluster is obviously an axis-parallel subspace cluster. Projecting the contributing points onto the contributing attributes, the points cluster at one single point. However, this single point is a special case, since it has identical attribute values in all directions and, hence, it is always located on the bisecting line of the subspace relevant to the cluster (cf. Figure 5.1).

For the following categories, we will focus on *perfect* biclusters. Generally, however, real-world biclusters cannot be expected to be *perfect*, the model will rather apply only approximately on the data. Allowing for imprecise models makes the task of finding biclusters even harder. One has to decide, when a cluster satisfactorily suffices the model in its general form. For example, optimizing for *perfect* constant biclusters on the matrix $\boldsymbol{A} = (X, Y)$ will probably lead to $|X| \cdot |Y|$ biclusters, each consisting of only one point and one dimension. How to avoid this kind of overfitting constitutes one interesting question of different contributions to this task.

5.1.2 Biclusters with Constant Values on Rows or Columns

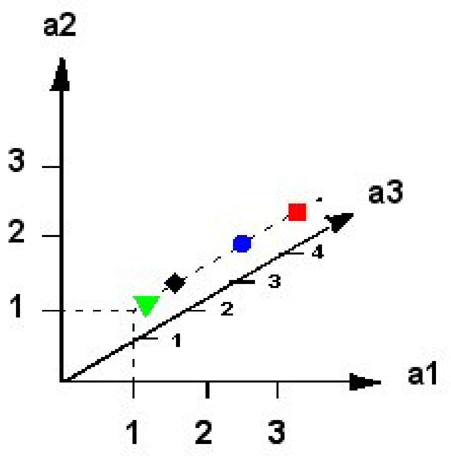
Biclusters with Constant Values on Columns.

Biclusters of points sharing constant values on columns are a more relaxed case of axis-parallel subspace clusters. The projection onto the contributing

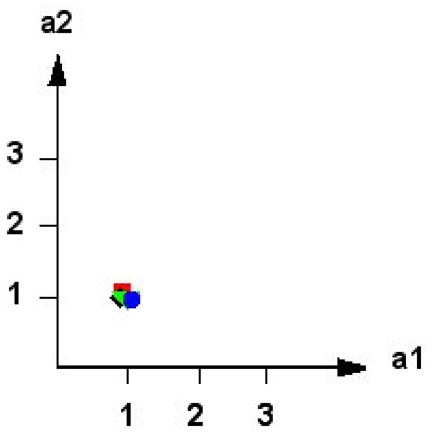


	a1	a2	a3
P1	1	1	3.5
P2	1	1	2.3
P3	1	1	0.2
P4	1	1	0.7

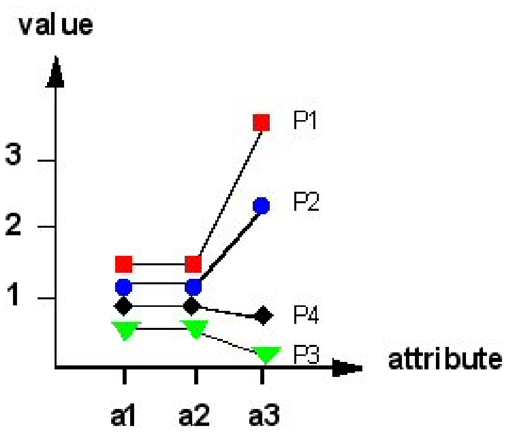
(a) Data matrix $\boldsymbol{A}_{XY}$



(b) Data space



(c) Subspace $\{a_1, a_2\}$



(d) Pattern

Figure 5.1: Constant bicluster

attributes yields once again a single point, but this point can be arbitrarily located anywhere in the corresponding subspace (cf. Figure 5.2). For the corresponding submatrix $\boldsymbol{A}_{IJ} = (I, J)$ it holds for a constant value μ which is typical for the cluster, with an adjustment value c_j for column $j \in J$, and for all $i \in I$ and $j \in J$:

$$a_{ij} = \mu + c_j. \tag{5.8}$$

Biclusters with Constant Values on Rows.

Biclusters with constant values on rows accommodate the participating points on the bisecting line of the participating dimensions (cf. Figure 5.3). For the corresponding submatrix $\boldsymbol{A}_{IJ} = (I, J)$ it holds for a constant value μ which is typical for the cluster, with an adjustment value r_i for row $i \in I$, and for all $i \in I$ and $j \in J$:

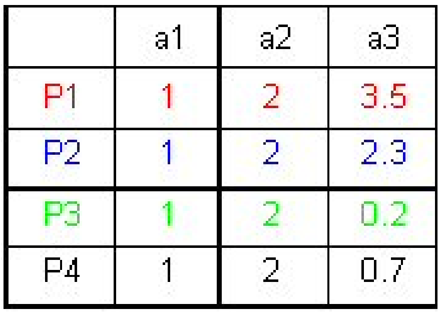
$$a_{ij} = \mu + r_i. \tag{5.9}$$

5.1.3 Biclusters with Coherent Values

More sophisticated approaches seek biclusters with coherent values exhibiting a particular form of covariance between rows and columns. One way to describe such biclusters is a combination of Equations 5.8 and 5.9. For a *perfect* bicluster with coherent values, (I, J), the values a_{ij} can be predicted by an additive model as

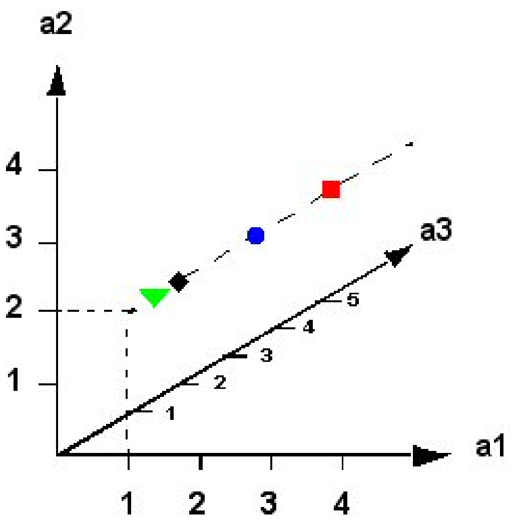
$$a_{ij} = \mu + r_i + c_j. \tag{5.10}$$

Again, r_i is an adjustment value for row $i \in I$, c_j is an adjustment value for column $j \in J$. The difference to the simpler model of constant values in rows or columns is constituted by using both adjustment values simultaneously to adjust the mean value μ to a certain value in row i and column j. In fact, biclusters with constant values in columns or rows, respectively, could

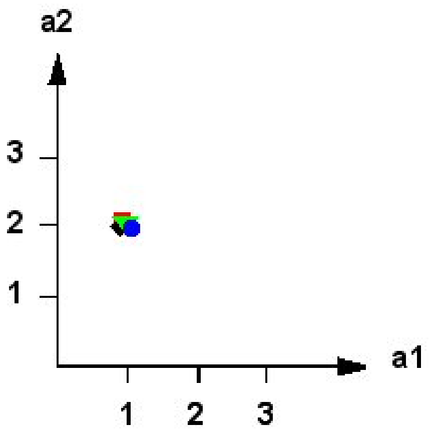


	a1	a2	a3
P1	1	2	3.5
P2	1	2	2.3
P3	1	2	0.2
P4	1	2	0.7

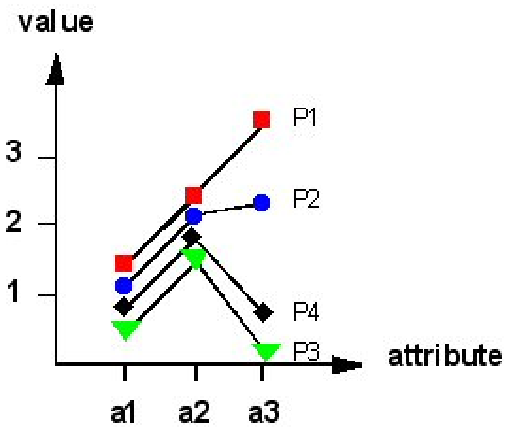
(a) Data matrix $\boldsymbol{A}_{XY}$



(b) Data space

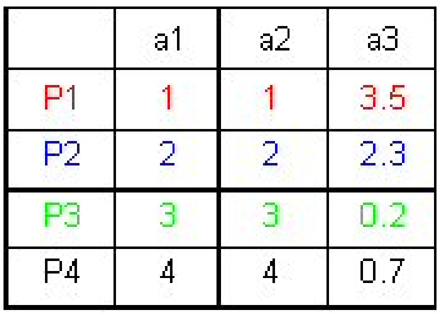


(c) Subspace $\{a_1, a_2\}$



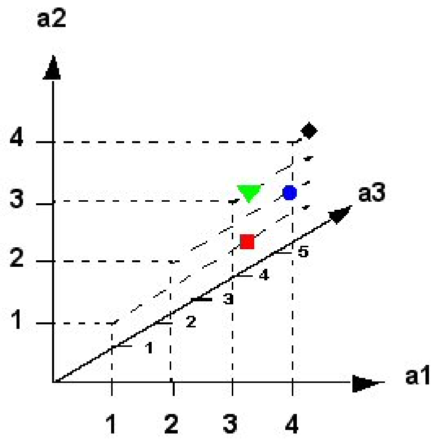
(d) Pattern

Figure 5.2: Constant values on columns

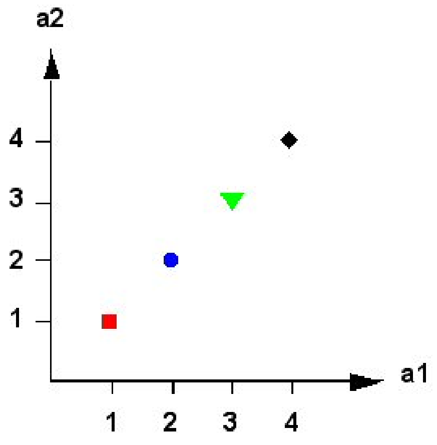


	a1	a2	a3
P1	1	1	3.5
P2	2	2	2.3
P3	3	3	0.2
P4	4	4	0.7

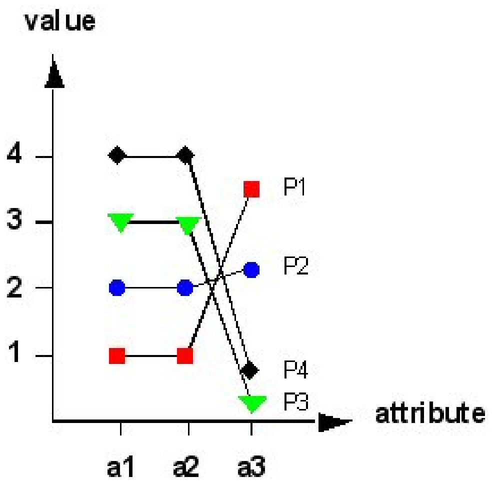
(a) Data matrix $\boldsymbol{A_{XY}}$



(b) Data space



(c) Subspace $\{a_1, a_2\}$



(d) Pattern

Figure 5.3: Constant values on rows

be regarded as special cases of biclusters with coherent values, where the adjustment values are $r_i = 0$ (resulting in Equation 5.8) or $c_j = 0$ (resulting in Equation 5.9).

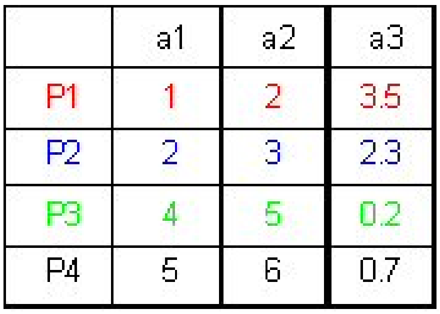
The corresponding clusters accommodate data points on hyperplanes parallel to the axes of irrelevant attributes in the complete data space. Projected onto the corresponding subspace, the clusters appear as increasing one-dimensional lines (cf. Figure 5.4). This pattern includes constant lines, which reduces to the special case of the category of biclusters with constant values on columns.

Note that decreasing lines are not covered by this model because those would result in a completely different pattern in the data matrix: while increasing lines consist of positively correlated attributes, decreasing lines result from negatively correlated attributes. The corresponding pattern cannot be described by the simple additive models typical for biclustering approaches which only cover shifted patterns (cf. Figure 5.5).

5.1.4 Biclusters with Coherent Evolutions

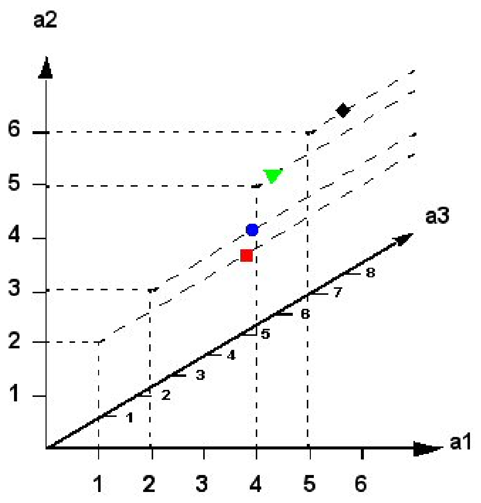
In this category, biclusters are constituted by a set of rows and columns, where the changes of attribute values are common among attribute pairs for all participating rows not in the exact quantity, but only in the fact, that a change happens at all. Some approaches require the change of attribute values to exhibit the same direction (either increasing or decreasing). Some approaches quantize the occurring attribute values in some discrete states and address equal state-transitions (e.g. [104, 131]). To obtain an intuition behind such bicluster models, imagine a quantizing approach with some states. The set of states constitutes a grid in the data space. A bicluster then contains a set of points I that fill the same grid cell in the projection of the set of attributes J contributing to the bicluster (cf. Figure 5.6).

Since quantizing approaches could bluntly be regarded as grid-based, axis-parallel subspace clustering (cf. Figure 5.6(c)), we will rather inspect an approach seeking clusters exhibiting a general tendency in attribute values as an

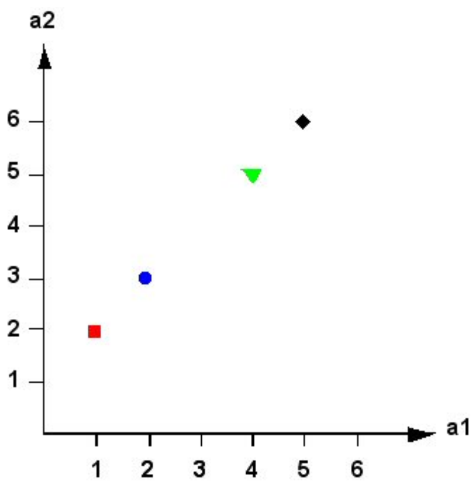


	a1	a2	a3
P1	1	2	3.5
P2	2	3	2.3
P3	4	5	0.2
P4	5	6	0.7

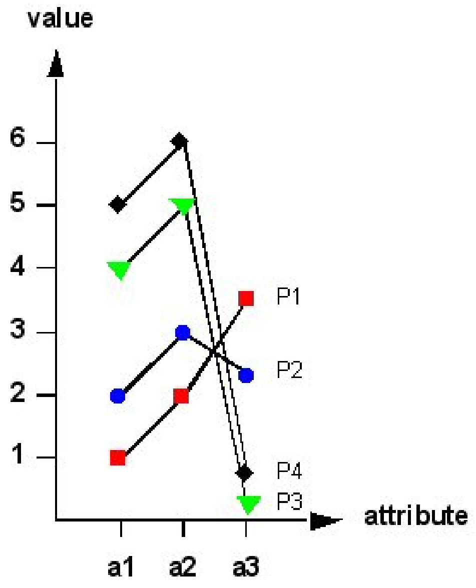
(a) Data matrix $\boldsymbol{A_{XY}}$



(b) Data space

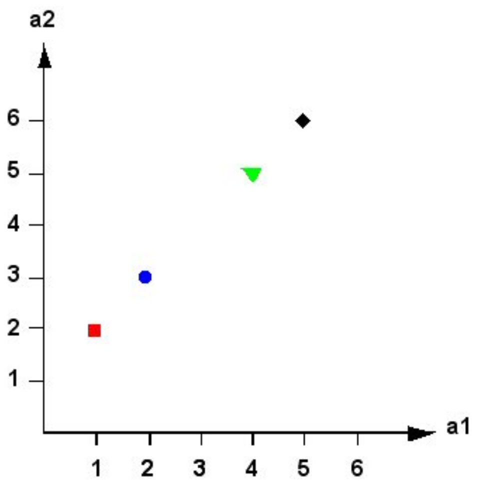


(c) Subspace $\{a_1, a_2\}$

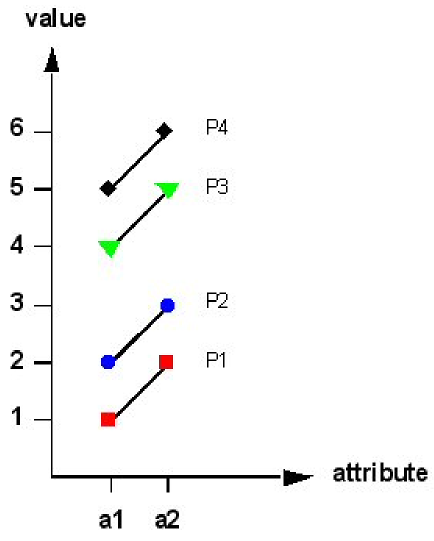


(d) Pattern

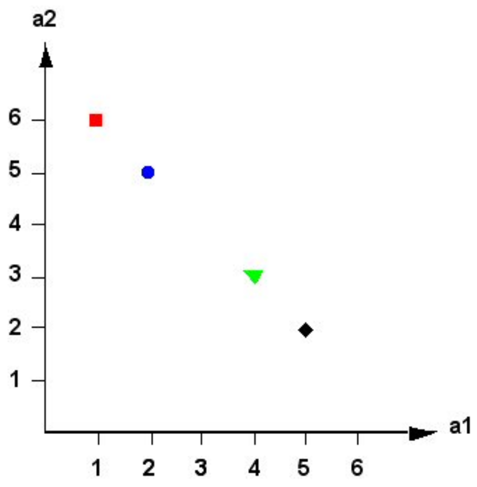
Figure 5.4: Coherent values



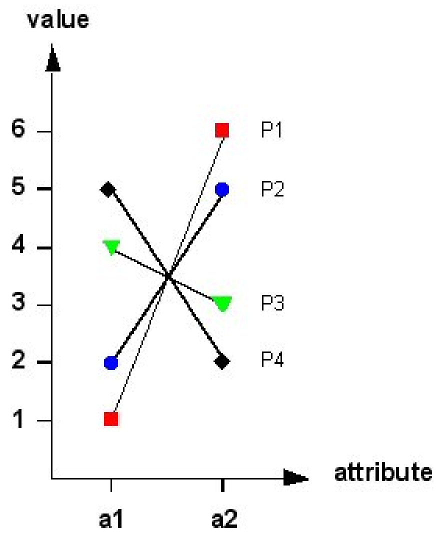
(a) Positive correlation



(b) Pattern



(c) Negative correlation

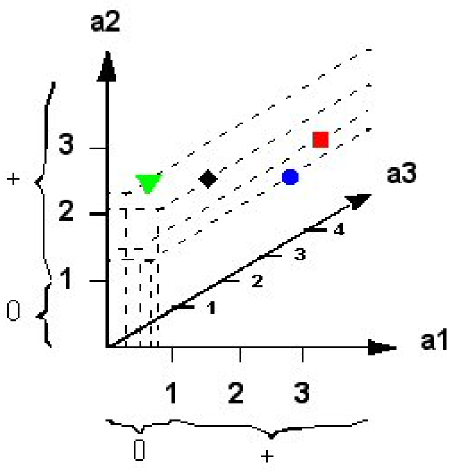


(d) Pattern

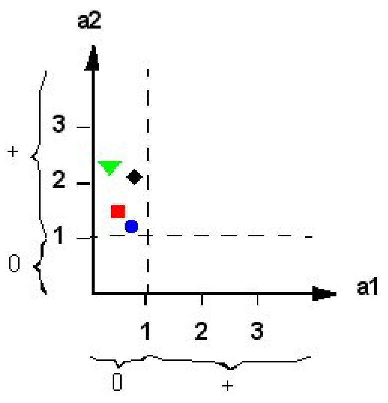
Figure 5.5: Patterns corresponding to positively and negatively correlated attributes

	a1	a2	a3
P1	0.5	1.5	3.5
P2	0.7	1.3	2.3
P3	0.3	2.3	0.2
P4	0.8	2.1	0.7

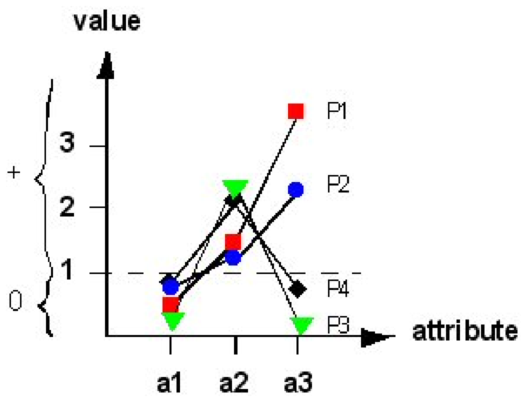
(a) Data matrix $\boldsymbol{A_{XY}}$



(b) Data space

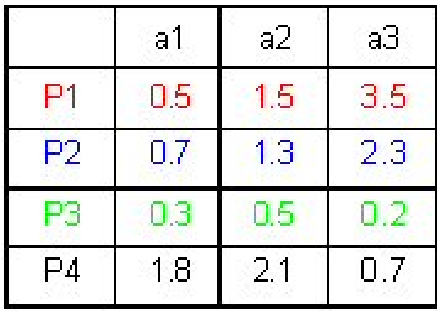


(c) Subspace $\{a_1, a_2\}$



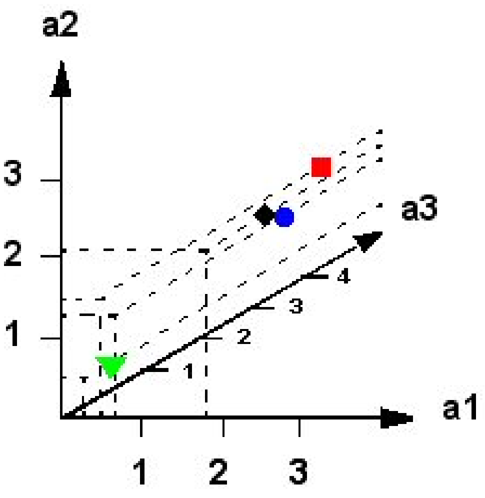
(d) Pattern

Figure 5.6: Coherent evolutions: state transitions

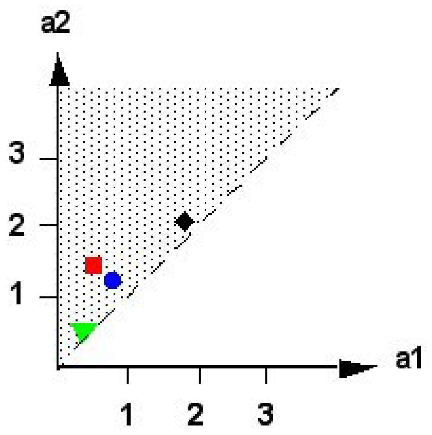


	a1	a2	a3
P1	0.5	1.5	3.5
P2	0.7	1.3	2.3
P3	0.3	0.5	0.2
P4	1.8	2.1	0.7

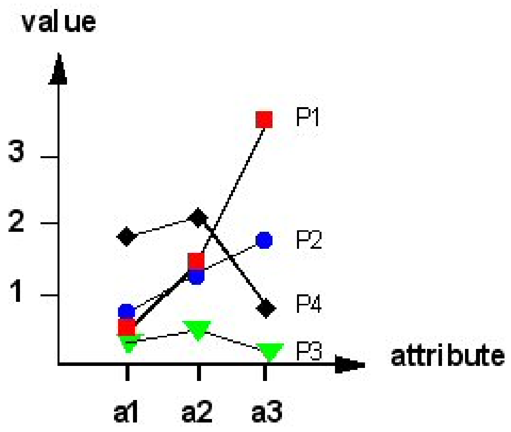
(a) Data matrix $\boldsymbol{A}_{XY}$



(b) Data space



(c) Subspace $\{a_1, a_2\}$



(d) Pattern

Figure 5.7: Coherent evolutions: change in the same direction

example for biclustering with coherent evolution patterns. This phenomenon has been grasped as the "order-preserving submatrix" problem [26]. The idea is to find a subset of rows and columns where a permutation of the set of columns exists such that the values in every row are increasing.

For this model, we find no spatial intuition corresponding to the visualizations given so far (cf. Figure 5.7). However, since all points are located in a half-space of the relevant subspace (cf. Figure 5.7(c)), one may probably find related approaches in the field of quantitative association rule mining (cf. [139, 118, 56]) or in the adaptation of formal concept analysis [54] to numeric data [112].

5.2 Pattern Based Clustering Algorithms

5.2.1 Constant Biclusters

Hartigan [66] provided the classical description of the biclustering problem. The quality of a bicluster is given by the sum of squares of all entries assuming the average value to form the corresponding ideal (*perfect*) bicluster. This could also be regarded as the variance of the submatrix $\boldsymbol{A}_{IJ}$, given the mean a_{IJ} (see Equation 5.3):

$$VAR(\boldsymbol{A}_{IJ}) = \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2 . \tag{5.11}$$

The data matrix is split recursively into two partitions. At each step, the split maximizing the reduction in the overall sum of squares of all biclusters is chosen. The splitting stops when the reduction of the sum of squares is less than that expected by chance.

This procedure is similar to a divisive, top-down hierarchical clustering and, therefore, results in a rather inefficient procedure.

5.2.2 Biclusters with Constant Values in Rows or Columns

Algorithms of this category usually apply a normalization to transform the biclusters into constant biclusters (e.g. [57]). Other approaches described in the bioinformatics community consider the existence of multiplicative noise, or constrain the values in rows and columns to certain intervals, or even provide probabilistic models for the clusters (e.g. [38, 125, 123]).

Besides these application driven methods, we know of no general biclustering approach to this specific problem. However, the problem seems not that intriguing after all, since it can easily be reduced to the first category, and, in turn, the following category treats problems belonging to this category as special cases.

Let us note that the general subspace and projected clustering algorithms described in Section 4 tackle the problem of finding biclusters with constant values on columns in a general way. The problem of finding biclusters with constant values on rows is a very special case of general correlation clustering algorithms (cf. Section 6).

5.2.3 Biclusters with Coherent Values

Cheng and Church [41] are credited with having introduced the term *biclustering* (inspired by [100]) to the analysis of gene expression (microarray) data. They assess the quality of a bicluster (I, J) by a *mean squared residue* value H given by

$$H(I,J) \quad = \quad \frac{1}{|I| \cdot |J|} \sum_{i \in I, j \in J} \left(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}\right)^2 . \tag{5.12}$$

The submatrix (I, J) is then considered a δ-bicluster if $H(I, J) \leq \delta$ for a given $\delta \in \mathbb{R}_0^+$. Setting $\delta = 0$ results in *perfect* δ-biclusters. In this model, a bicluster is *perfect*, if each row and column exhibits an absolutely consistent bias. The bias of column j w.r.t. the other columns is given by $a_{Ij} - a_{IJ}$. The bias of row i w.r.t. the other rows is given by $a_{iJ} - a_{IJ}$. In a *perfect* δ-bicluster, the value a_{ij} is then given additively by a row-constant, a column-constant, and an overall constant value:

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ}. \tag{5.13}$$

Setting $\mu = a_{IJ}$, $r_i = a_{iJ} - a_{IJ}$, and $c_j = a_{Ij} - a_{IJ}$, this model corresponds to the general description of additive models for biclusters with coherent values given by Equation 5.10. However, the value a_{ij} is not directly given as in Equation 5.13 whenever a δ-bicluster is *not* perfect. In this case, the value predicted by the model will deviate from the true model:

$$a_{ij} = res(a_{ij}) + a_{iJ} + a_{Ij} - a_{IJ}. \tag{5.14}$$

Equivalently, the residue is given by

$$res(a_{ij}) = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}. \tag{5.15}$$

This value is used in Equation 5.12 to calculate the mean squared residue. In a similar way, the mean squared residue of a row i or of a column j, respectively, are defined as

$$d(i) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2, \tag{5.16}$$

$$d(j) = \frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2. \tag{5.17}$$

To find a δ-bicluster, Cheng and Church propose to greedily remove the row or column (or a set of rows or columns) with maximal mean squared residue of the row or the column until the remaining submatrix (I, J) satisfies $H(I, J) \leq \delta$. Afterwards, in order to find a *maximal* δ-bicluster, rows and columns are added to (I, J) unless adding the row or column would increase the value $H(I, J)$. Curiously, if the number of rows or columns is below 100 (which is still a pretty high number of objects or a high dimensionality, respectively, in many applications), no multiple row or column deletion is performed.

So far, the procedure finds one δ-bicluster in a data matrix. In order to find k δ-biclusters, the procedure is iterated k times. Any δ-bicluster already found has to be masked by random numbers. This makes it unlikely that elements covered by existing biclusters would contribute to any future bicluster, but does not remove complete rows and columns. The finally resulting set of k biclusters should therefore be disjunct w.r.t. combinations of rows and columns, i.e., a data point cannot contribute to different clusters based on the same attributes, and an attribute cannot contribute to different clusters for the same data point. Thus, in our sense, the clusters may overlap, and the subspaces may overlap, but, in theory, not both at the same time. Besides inserting random numbers, the algorithm is deterministic and should retrieve equal results in different runs. If not, the random numbers contributed to a cluster, which would be an alarming effect.

Several later contributions to the field are based on the δ-bicluster model as proposed by Cheng and Church but address some issues bequeathed by their approach. As weak points[2] in the approach of Cheng and Church one could state:

1. One cluster at a time is found, then the cluster needs to be masked in order to find a second cluster.
2. This procedure bears an inefficient performance.
3. The masking may lead to less accurate results.
4. The masking inhibits simultaneous overlapping of rows and columns.
5. Missing values cannot be dealt with.
6. The user must specify the number of clusters beforehand.

With FLOC, Yang et al. [144] introduce another algorithm to find δ-biclusters. FLOC is a randomized move-based algorithm efficiently approximating k δ-clusters, again based on the minimization of the average residue. Initial seed-clusters are optimized by randomly chosen steps of removing or

[2]In some cases, however, whether a certain point is a “weak” point is a matter of taste. The stated weak points are addressed as such in several publications reported below.

adding a row or a column. This addresses issue 1. Furthermore, the algorithm allows simultaneous overlapping of rows and columns (issue 4). The model is adapted in taking only specified values into account for the computation of the residue, thus allowing for missing values, addressing issue 5.[3]

However, the improvements stated in [144] are paid for by introducing random events. Therefore, the same authors propose also a deterministic approach with the p-cluster model [138]. This model specializes the δ-bicluster-property to a pairwise property of two objects on two attributes as

$$|(a_{i_1j_1} - a_{i_1j_2}) - (a_{i_2j_1} - a_{i_2j_2})| \leq \delta \tag{5.18}$$

or equivalently

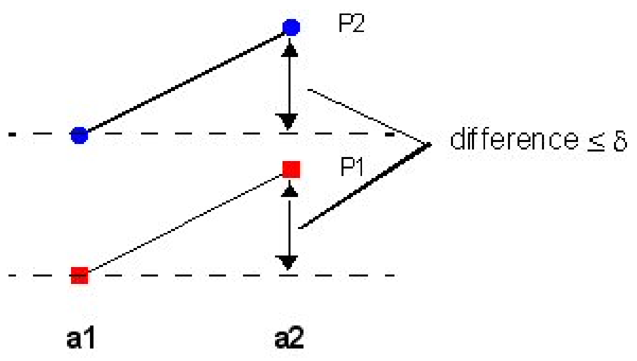
$$|(a_{i_1j_1} - a_{i_2j_1}) - (a_{i_1j_2} - a_{i_2j_2})| \leq \delta. \tag{5.19}$$

Inequality 5.18 describes the difference between two objects by their relative differences of two attribute values (cf. Figure 5.8(a)). Inequality 5.19 describes the difference of two attributes by the absolute differences between two objects. Both conditions cover identical sets of cases. A submatrix (I, J) is a δ-p-cluster, if this property is fulfilled for any 2×2-submatrix $(\{i_1, i_2\}, \{j_1, j_2\})$, where $\{i_1, i_2\} \subseteq I$ and $\{j_1, j_2\} \subseteq J$. Formulating the pattern description as a pairwise condition tightens the model for biclusters. While limiting the overall variance of a bicluster may allow to include some outliers in a cluster, the pairwise condition excludes outliers more rigorously (cf. Figure 5.8(c)).

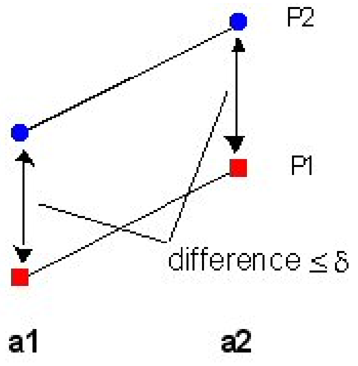
After creating the maximal set of attributes for each pair of objects forming a δ-p-cluster and the maximal set of objects for each pair of attributes forming a δ-p-cluster, a pruning-step is implemented to lower the impact of the final step, the search in the set of submatrices. This search, however, requires exponential time after all. Addressed issues w.r.t. the Cheng and Church approach are 1, 4, and 6.

Another related approach is MaPle [110], stressing the maximality of the mined δ-p-clusters, based on the closure property of subclusters coming along

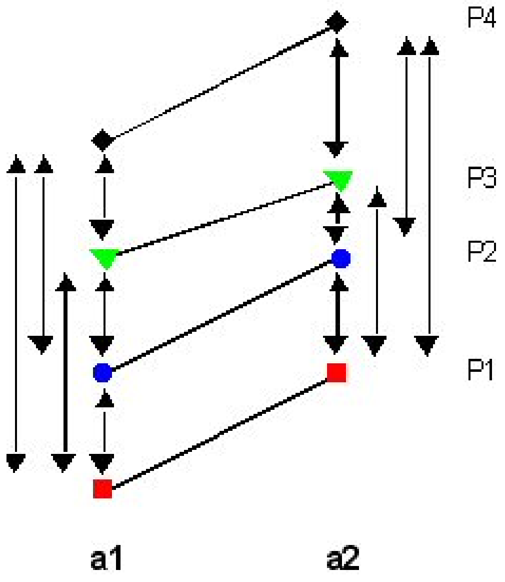
[3]For another algorithm allowing for missing values see [126].



(a) Visualization of Inequality 5.18



(b) Visualization of Inequality 5.19



(c) Pairwise differences uncover outliers

Figure 5.8: p-cluster model: pairwise differences

with the model of δ-p-clusters: For a δ-p-cluster (I, J), every submatrix (I', J') with $I' \subseteq I$ and $J' \subseteq J$ is a δ-p-cluster. Note that this is not necessarily true for the more general bicluster models of Cheng and Church [41] or Yang et al. [144], since an outlier may be covered by a bigger cluster but would influence the variance of a smaller cluster considerably (cf. Figure 5.8(c)). MaPle mines for maximal δ-p-clusters, i.e. for a given δ-p-cluster (I, J) there exists no other δ-p-cluster (M, N) in the data set with $I \subset M$ and $J \subset N$. Essentially, MaPle performs the analysis in a similar way as the previous approach [138], but uses the closure property for pruning any superset D' of the attribute set D once D is found unsuitable to serve as base for a p-cluster. Still, like the approach in [138], also MaPle is based on a complete enumeration in the end. Thus, once again, the addressed issues of the Cheng and Church approach include items 1, 4, and 6.

The CoClus algorithms proposed by Cho et al. [42] seek a marriage of a k-means-like approach with the cluster models of Hartigan or Cheng and Church. To avoid poor local minima and empty clusters, a local search strategy is implemented swapping single rows between clusters if this reduces the objective function. The addressed issues w.r.t. the Cheng and Church approach are therefore 1 and 2. However, these algorithms cannot avoid the typical flaws of k-means-like approaches as being caught in local minima (despite the local search strategy), requiring specification of the number k of clusters beforehand, and, as a complete partition of the data set onto k clusters that are disjunct w.r.t. rows as well as columns, the data set is assumed to contain no noise. Instead, every attribute is assumed to be relevant for exactly one cluster. This assumption generally contradicts the circumstances of clustering high dimensional data and the presence of noise will generally deteriorate the quality of the clustering result.

5.2.4 Biclusters with Coherent Evolutions

The concept of an order preserving submatrix (OPSM), introduced by Ben-Dor et al. [26], describes a submatrix (I, J) of the data matrix $\boldsymbol{A}$ where a permutation π of the set of columns J exists such that for each row $i \in I$ and each index $1 \leq m < |J|$ within the permutation $\pi(J)$ the following inequality holds:

$$a_{i\pi(J)[m]} < a_{i\pi(J)[m+1]}, \tag{5.20}$$

i.e., according to the given linear order of columns the values in the selected columns are strictly increasing. The cluster model is then given by the pair (J, π). The *support* of a model is the set I of rows fitting to the model according to Inequality 5.20.

The algorithm of Ben-Dor et al. searches the best model in a greedy bottom-up-approach (i.e., starting with small models and iteratively extending the best l of these models). The "best" model is the one with largest statistical significance (i.e., having the smallest prior probability). This algorithm favors models with a large support.

Liu and Wang [96] follow the same general idea defining a bicluster as OP-Cluster (order preserving cluster) but weaken the conditions of Ben-Dor et al. by introducing groups of similar attributes. They also discard the assessment of statistical significance and instead report all (maximal) submatrices covering at least a given minimum number of rows and columns. The algorithm creates a non-decreasing order of columns for each row (grouping together similar columns). The resulting set of column-sequences is mined for frequent patterns.

As said above, we do not have any spatial intuition explaining this model. The resulting biclusters consist of objects showing a similar trend on a set of attributes. Whether the corresponding attributes are correlated or even linearly correlated remains unclear. The results may be interesting in some application domains, but the clusters are not necessarily accommodated in any specific subspace of the data space. Considering the model as sketched above (cf. Inequality 5.20), clearly there is no way to predict an attribute value for a given instance and a specified column. The model merely allows to predict the value to exceed a given threshold, namely the attribute value in the preceding column. Hence, the points occupy half-spaces and, as stated above, we find this approach being related to quantitative association rule mining.

5.3 Summary

Independent of the concrete problem formulation and the spatial intuition behind it, the biclustering problem is sometimes formulated as a graph mining problem (e.g. [43, 131, 126]). The data matrix is then described as a bipartite graph, where one set of vertices corresponds to the rows, the other set corresponds to the columns. As an example problem formulation, finding a minimum set of biclusters to cover all the elements in a data matrix is a generalization of the problem of covering a bipartite graph by a minimum set of bicliques, a problem known to be NP-hard [55]. The exact complexity of a biclustering problem depends on the exact problem definition and the merit function used to evaluate the quality of a specific biclustering. For most of the common biclustering problems the computational complexity is not known. If, however, the computational complexity of a specific biclustering problem-formulation is known, it usually is an NP-hard problem. Thus, different heuristics, simplifying models, and greedy or randomized approaches are implemented.[4]

Although biclustering models do not fit exactly into the spatial intuition behind subspace, projected, or correlation clustering (a summarizing comparison of patterns in the data matrix, corresponding bicluster models, and related spatial patterns is given in Figure 5.9), these models make sense in view of a data matrix and fruitful applications seem to justify the approach. However, since the cluster models forming the basis of biclustering algorithms differ considerably, conducting a fair comparison among these algorithms is a non-trivial task. A thorough evaluation — let alone in comparison with axis-parallel and correlation clustering algorithms — is not available in the literature. Recently, Prelić et al. [113] performed an evaluation of five selected methods [41, 131, 26, 73, 104]. However, a thorough comparison of clustering-algorithms requires quite some effort since there is not the solely and uniquely established comparison method of clusterings.

[4]Let us note that, of course, clustering in general is also an NP-hard problem although we are used to efficient solutions. Every efficient clustering algorithm can thus only provide an approximative solution based on certain assumptions and heuristics, e.g. reflected by its underlying cluster model.

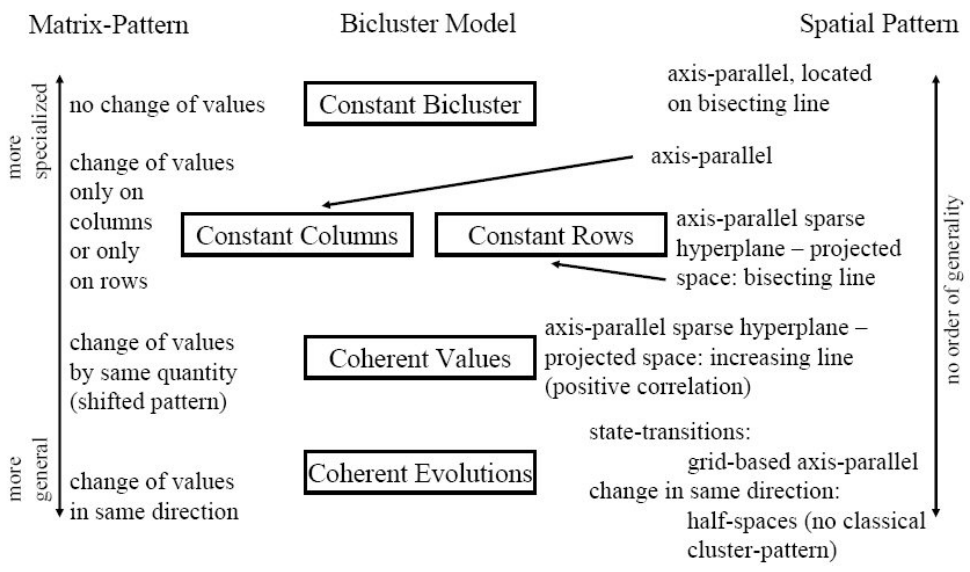

Matrix-Pattern
Bicluster Model
Spatial Pattern
more specialized
more general
no change of values
Constant Bicluster
axis-parallel, located on bisecting line
change of values only on columns or only on rows
axis-parallel
Constant Columns
Constant Rows
axis-parallel sparse hyperplane – projected space: bisecting line
change of values by same quantity (shifted pattern)
Coherent Values
axis-parallel sparse hyperplane – projected space: increasing line (positive correlation)
change of values in same direction
Coherent Evolutions
state-transitions: grid-based axis-parallel
change in same direction: half-spaces (no classical cluster-pattern)
no order of generality

Figure 5.9: Comparison: Patterns in Biclustering approaches and their corresponding spatial intuitions

In view of the rather specialized tasks performed by biclustering approaches, the comparison of performance w.r.t. effectiveness as well as efficiency should be performed in a broad context of subspace clustering, projected clustering, and correlation clustering. Despite (or by virtue) of the specialization of the biclustering cluster models, this family of clustering approaches seems to perform well on microarray data. In this application domain, there exists a vast amount of approaches not covered in this survey (cf. several surveys focussed on biological and medical applications [137, 75, 97, 132]).

Chapter 6

Finding Clusters in Arbitrarily Oriented Subspaces

6.1 General Aim of Algorithms in this Category

While the aim of pattern-based clustering algorithms is easily understood in terms of corresponding matrix representations, the spatial intuition behind these approaches is not quite so convincing. A clear pattern in a matrix corresponds to special or even rather artificial constellations of the corresponding points in space (or even no specific constellation at all). The explaining models remain rather simple and cannot include simple negative correlations let alone more complex correlations.

A more general, intuitive approach is adopted by a family of algorithms known as *oriented clustering* or *generalized subspace/projected clustering* or *correlation clustering*[1] algorithms. These algorithms assume any cluster being located in an arbitrarily oriented subspace S of the data space $\mathbb{R}^d$. Such

[1]Note that the term "correlation clustering" relates to a different task in the machine learning community, where a partitioning of the data shall correlate as much as possible with a pairwise similarity function f learned from past data (e.g. cf. [20]).

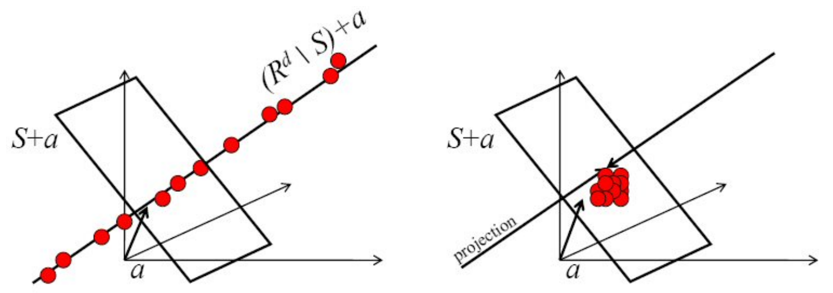


Figure 6.1: Points distributed in data space where some attributes are correlated cluster densely in a certain projection (arbitrarily oriented subspace).

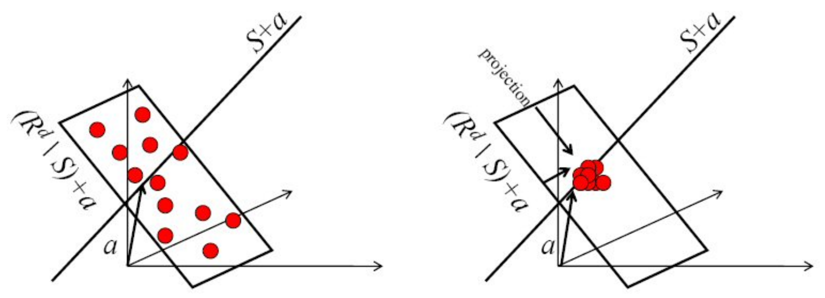


Figure 6.2: Points distributed in an arbitrarily oriented subspace form a hyperplane.

clusters appear as hyperplanes of arbitrary dimensionality in the data space. However, no typical pattern in the data matrix does correspond to these models based on a spatial intuition. Thus, all approaches known from biclustering need to be dismissed, albeit some bicluster models can be regarded as special cases of correlation clustering models.

In terms of subspace clustering, an affine subspace $S + \vec{a}$, $S \subset \mathbb{R}^d$ with affinity $\vec{a} \in \mathbb{R}^d$ is interesting if a set of points exhibits a certain density within this subspace (i.e., projected onto this subspace) regardless of high variances along axes in the perpendicular subspace $(\mathbb{R}^d \setminus S) + \vec{a}$. In *oriented clustering*, these interesting subspaces need not be axis-parallel, but can be arbitrarily oriented (cf. Figure 6.1).

Assuming high variance along those axes forming the perpendicular subspace $(\mathbb{R}^d \setminus S) + \vec{a}$, the points building the cluster in S fill the perpendicular subspace $(\mathbb{R}^d \setminus S) + \vec{a})$ to a certain extension. In the complete data space $\mathbb{R}^d$, the points therefore form a hyperplane which is located in the subspace $(\mathbb{R}^d \setminus S) + \vec{a}$ (cf. Figure 6.2).

Nevertheless, this observation facilitates an alternative way of describing *oriented clustering*. Points accommodated on a common hyperplane in data space appear to follow linear dependencies among the attributes participating in the description of the hyperplane. Since linear dependencies result in the observation of strong linear correlations among these attributes, we call this type of clustering also "*correlation clustering*".

In fact, describing correlation clusters in terms of a subspace (hyperplane) accommodating the points rather than in terms of the subspace, where the points show high density, opens up different possibilities. Consider again the simple problem introduction in Figure 3.2. If some arbitrarily oriented subspaces are chosen to project the points and to search for dense sets of points, in this example, again all points would cluster densely in all three of the chosen subspaces. However, the points are accommodated on different hyperplanes orthogonal to the chosen subspaces.

A common technique to grasp arbitrarily oriented directions of high variance in a data set is Principal Component Analysis (PCA — for a thorough introduction see [79]). Again, a local application of PCA is required in order to find clusters located in different subspaces (cf. the general problem statement in Chapter 3). Generally, applying PCA to a local selection of points is again based on the locality assumption. It is assumed that the hyperplane accommodating the points of a correlation cluster is sufficiently reflected in a local selection of points (e.g. the ε-neighborhood or the k nearest neighbors of a point).

As a general idea, assume we apply a PCA-based approach to a given set of points $\mathcal{D} \subset \mathbb{R}^d$ in order to find the directions of high and low variance. First, we build the covariance matrix $\boldsymbol{\Sigma}_\mathcal{D}$ of $\mathcal{D}$:

$$\boldsymbol{\Sigma}_\mathcal{D} \;=\; \frac{1}{|\mathcal{D}|} \cdot \sum_{\vec{x} \in \mathcal{D}} (\vec{x} - \vec{x}_\mathcal{D}) \cdot (\vec{x} - \vec{x}_\mathcal{D})^\mathsf{T}, \tag{6.1}$$

where $\vec{x}_\mathcal{D}$ denotes the centroid (mean) of all points $\vec{x} \in \mathcal{D}$. $\boldsymbol{\Sigma}_\mathcal{D}$ is a $d \times d$

symmetric positive semidefinite matrix where $\sigma_{\mathcal{D}ij}$ (i.e., the value at row i and column j in $\boldsymbol{\Sigma}_{\mathcal{D}}$) equals the covariance between the dimensions i and j. The diagonal entry $\sigma_{\mathcal{D}ii}$ corresponds to the variance of the ith dimension. $\boldsymbol{\Sigma}_{\mathcal{D}}$ can be decomposed (by PCA[2]) into the *eigenvalue* matrix $\boldsymbol{E}_{\mathcal{D}}$ of $\boldsymbol{\Sigma}_{\mathcal{D}}$ and the *eigenvector* matrix $\boldsymbol{V}_{\mathcal{D}}$ of $\boldsymbol{\Sigma}_{\mathcal{D}}$ such that

$$\boldsymbol{\Sigma}_{\mathcal{D}} = \boldsymbol{V}_{\mathcal{D}} \cdot \boldsymbol{E}_{\mathcal{D}} \cdot \boldsymbol{V}_{\mathcal{D}}^{\mathsf{T}}. \tag{6.2}$$

The eigenvalue matrix $\boldsymbol{E}_{\mathcal{D}}$ is a diagonal matrix holding the eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{D}}$ in decreasing order in its diagonal elements. The eigenvector matrix $\boldsymbol{V}_{\mathcal{D}}$ is an orthonormal matrix with the eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{D}}$ ordered correspondingly to the eigenvalues in $\boldsymbol{E}_{\mathcal{D}}$. The eigenvectors provide a new orthonormal basis. The eigenvalue matrix $\boldsymbol{E}_{\mathcal{D}}$ can be understood as the covariance matrix of the original data set when represented in the new axis system $\boldsymbol{V}_{\mathcal{D}}$. All non-diagonal entries equal zero meaning that all covariances have been removed. The first eigenvector in $\boldsymbol{V}_{\mathcal{D}}$ points to the direction of the highest variance in the data set $\mathcal{D}$. The second eigenvector points to the direction of the second highest variance in $\mathcal{D}$ perpendicular to the first eigenvector. Assuming $\mathcal{D}$ being a correlation cluster, the major axes in $\boldsymbol{V}_{\mathcal{D}}$, say the first λ eigenvectors, span the λ-dimensional hyperplane accommodating the points of $\mathcal{D}$. We denote the first λ eigenvectors of $\boldsymbol{V}_{\mathcal{D}}$ by $\check{\boldsymbol{V}}_{\mathcal{D}}$ and call them *strong* eigenvectors. The remaining eigenvectors are called *weak* eigenvectors (denoted by $\hat{\boldsymbol{V}}_{\mathcal{D}}$). The weak eigenvectors can equivalently define the hyperplane accommodating the points of $\mathcal{D}$ as they all are orthogonal to that hyperplane. While the trace $\sum_{i=1}^{d} \sigma_{\mathcal{D}ii}$ is invariant under the axis transformation defined by the eigensystem $\boldsymbol{V}_{\mathcal{D}}$ (i.e., $\sum_{i=1}^{d} \sigma_{\mathcal{D}ii} = \sum_{i=1}^{d} e_{\mathcal{D}ii}$), the sum of the smallest $d - \lambda$ eigenvalues $\sum_{i=\lambda+1}^{d} e_{\mathcal{D}ii}$ is the minimum under all possible transformations. Thus, the smallest $d - \lambda$ eigenvectors define the subspace perpendicular to the hyperplane accommodating the cluster members, where the projected points would cluster optimally dense.

How to combine PCA (or related means to discern different subspaces) with the selection of interesting subspaces, with a suitable definition of λ, with the selection of points, and with distance measures mainly makes the

[2]This decomposition is generally in $O(d^3)$.

difference among the most common approaches to the task of correlation clustering.

What means soever are used to find correlation clusters, the general meaning of this family of clusters results in a mathematically clear, explanatorily rich, and predictively powerful model for correlation clusters [4]: the λ-dimensional hyperplane accommodating the points of a correlation cluster $\mathcal{C} \subset \mathbb{R}^d$ can be defined by a linear equation system consisting of $d - \lambda$ equations for d variables, and the affinity, e.g. given by the mean point $\vec{x}_\mathcal{C} = (\bar{x}_1 \cdots \bar{x}_d)^\mathsf{T}$ of all cluster members:

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

where v_{ij} is the value at row i, column j in the eigenvector matrix $\boldsymbol{V}_\mathcal{C}$ derived by PCA from the covariance matrix of $\mathcal{C}$. As introduced generally above, the first λ eigenvectors (i.e., the *strong* eigenvectors) give the directions of high variance and span the hyperplane accommodating $\mathcal{C}$. The remaining $d - \lambda$ *weak* eigenvectors span the perpendicular subspace. The linear equation system as sketched above can therefore be given by

$$\hat{\boldsymbol{V}}_\mathcal{C}^\mathsf{T} \cdot \vec{x} \;=\; \hat{\boldsymbol{V}}_\mathcal{C}^\mathsf{T} \cdot \vec{x}_\mathcal{C} \tag{6.3}$$

The defect of $\hat{\boldsymbol{V}}_\mathcal{C}^\mathsf{T}$ gives the number of free attributes, the remaining attributes may actually be involved in linear dependencies. The equation system is by construction at least approximately fulfilled for all points $\vec{x} \in \mathcal{C}$ and provides a quantitative model for the cluster.

However, from the user's point of view of the application of correlation clustering on a new data set, the correlations among attributes are observable, while the linear dependencies are merely an assumption to explain the correlations. Whether or not this assumption is valid can be evaluated by using the model as a predictive model and by refining the experiments based on the insights provided by the quantitative model (cf. [4]). Thus, this model of clustering is not only far more general than the models for biclustering

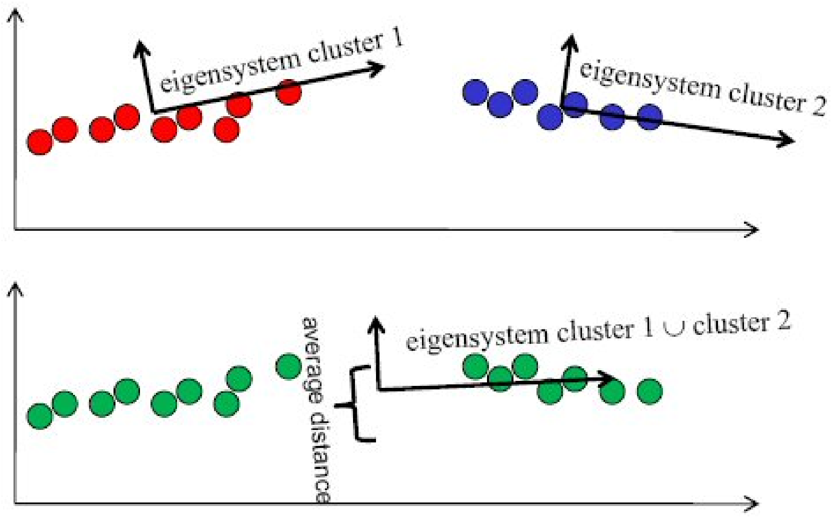


Figure 6.3: ORCLUS: distance of two clusters

sketched in Chapter 5 but also more concise and meaningful. The development of this model along with some applications constitutes a contribution of the author and is discussed in more detail in Part V.

6.2 Correlation Clustering Algorithms

6.2.1 PCA Based Approaches

The broad majority of correlation clustering approaches are based on an application of PCA on subsets of points (like range queries or k-nearest neighbor queries).

As the first approach to *generalized projected clustering*, Aggarwal and Yu [11] proposed the algorithm ORCLUS, using ideas similar to the axis-parallel approach PROCLUS [10]. ORCLUS is a k-means like approach, picking $k_c > k$ seeds at first, assigning the data base objects to these seeds according to a distance function that is based on an eigensystem of the corresponding cluster assessing the distance along the *weak* eigenvectors only (i.e., the distance in the projected subspace where the cluster objects exhibit high density).

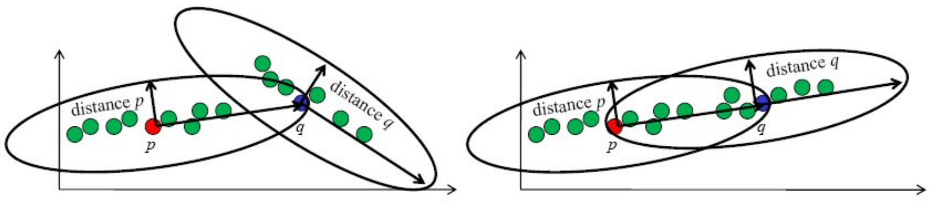


Figure 6.4: 4C: distance between two points

The eigensystem is iteratively adapted to the current state of the updated cluster. The number k_c of clusters is reduced iteratively by merging closest pairs of clusters until the user-specified number k is reached. The closest pair of clusters is the pair with the least average distance in the projected space (spanned by the weak eigenvectors) of the eigensystem of the merged clusters (cf. Figure 6.3). Starting with a higher k_c increases the effectiveness, but also the runtime. The method proposed in [39] is a slight variant of ORCLUS designed for enhancing multi-dimensional indexing. Another, presumably more efficient variant is proposed in [91].

In contrast to ORCLUS, the algorithm 4C [35] is based on a density-based clustering paradigm [47]. Thus, the number of clusters is not decided beforehand but clusters grow from a seed as long as a density criterion is fulfilled. Otherwise, another seed is picked to start a new cluster. The density criterion is a required minimal number of points within the neighborhood of a point, where the neighborhood is ascertained based on distance matrices computed from the eigensystems of two points. The eigensystem of a point $\vec{p}$ is based on the covariance matrix of the ε-neighborhood of $\vec{p}$ in Euclidean space. A parameter δ discerns large from small eigenvalues. In the eigenvalue matrix $\boldsymbol{E}_{\vec{p}}$ then large eigenvalues are replaced by 1, small eigenvalues by a value $\kappa \gg 1$. Using the adapted eigenvalue matrix $\boldsymbol{E'}_{\vec{p}}$, a correlation similarity matrix for $\vec{p}$ is obtained by $\boldsymbol{V}_{\vec{p}} \cdot \boldsymbol{E'}_{\vec{p}} \cdot \boldsymbol{V}_{\vec{p}}^{\mathsf{T}}$. This matrix is then used to derive the distance of two points, $\vec{q}$ and $\vec{p}$, w.r.t. $\vec{p}$, as the general quadratic form distance:

$$\sqrt{(\vec{p}-\vec{q})^{\mathsf{T}} \cdot \boldsymbol{V}_{\vec{p}} \cdot \boldsymbol{E'}_{\vec{p}} \cdot \boldsymbol{V}_{\vec{p}}^{\mathsf{T}} \cdot (\vec{p}-\vec{q})}. \tag{6.4}$$

Applying this measure symmetrically to $\vec{q}$ and choosing the maximum of both distances helps to decide whether both points are connected by a similar correlation of attributes and, thus, are similar and belong to each other's correlation neighborhood. Figure 6.4 illustrates this idea. The ellipsoids represent the correlation neighborhoods of some sample objects. In the left example of Figure 6.4, p and q are not connected because q does not find p in its correlation neighborhood. On the right hand side, the points p and q are connected because they find one another in their correlation neighborhood.

As a hierarchical approach, HiCO [7] defines the distance between points according to their local correlation dimensionality and subspace orientation and uses hierarchical density-based clustering [16] to derive a hierarchy of correlation clusters.

COPAC [6] is based on similar ideas as 4C but disposes of some problems like meaningless similarity matrices due to sparse ε-neighborhoods instead taking a fixed number k of neighbors — which raises the question how to choose a good value for k but at least choosing $k > \lambda$ ensures a meaningful definition of a λ-dimensional hyperplane. The main point in COPAC, however, is a considerable speed-up by partitioning the data set based on the observation that a correlation cluster should consist of points exhibiting the same local correlation dimensionality (i.e., the same number of strong eigenvectors in the covariance matrix of the k nearest neighbors). Thus, the search for clusters involves only the points with equal *local correlation dimensionality*. By creating one partition for each occurring correlation dimensionality, the time complexity rapidly decreases on average by getting rid of a squared factor d^2 in a d-dimensional data set.

Another related algorithm is ERiC [5], also deriving a local eigensystem for a point based on the k nearest neighbors in Euclidean space. Here, the neighborhood criterion for two points in a DBSCAN-like procedure is an approximate linear dependency and the affine distance of the correlation hyperplanes as defined by the strong eigenvectors of each point. Like in COPAC, the property of clusters to consist of points exhibiting an equal local correlation dimensionality is exploited for the sake of efficiency. Furthermore, the resulting set of clusters is also ordered hierarchically to provide the user with a hierarchy of subspace clusters. In finding and correctly assigning complex patterns of intersecting clusters, COPAC and ERiC improve considerably over ORCLUS and 4C.

Another approach based on PCA said to find even non-linear correlation clusters, CURLER [136], seems not restricted to correlations of attributes but, according to its restrictions, finds any narrow trajectory and does not provide a model describing its findings.

PCA is a mature technique and allows the construction of a broad range of similarity measures grasping local correlation of attributes and, therefore, to find arbitrarily oriented subspace clusters. A major intrinsic drawback common to all mentioned approaches is the notorious locality assumption. This assumption is widely accepted. But note that this innocent looking little (and often tacit) assumption boldly contradicts the basic problem statement: to find clusters in high dimensional space that is doomed by the curse of dimensionality. To address problems occurring due to varying density in the local neighborhood, a framework for selecting a suitable neighborhood range and to stabilize the PCA by weighting the points has been proposed in [90]. This framework allows to integrate all existing PCA-based correlation clustering approaches and shows considerable enhancements in effectiveness. However, this is not the ultimate solution for problems of high dimensional data spaces. As we will further discuss in more detail in Chapter 7, the curse of dimensionality condemns all distances to look alike and, thus, renders nearest neighbor queries rather meaningless in high dimensional data. Thus, to successfully employ PCA in correlation clustering in really high dimensional data spaces may require even more effort henceforth.

The algorithms 4C, COPAC, HiCO, and ERiC as well as the framework for stabilizing PCA-based approaches are contributions of the author and will be discussed in more detail in Part III.

6.2.2 An Approach Based on the Hough Transform

A completely different approach is pursued by the algorithm CASH [1] based on the concepts of the Hough transform [71, 45]. The Hough transform was originally designed to map the points from a 2-dimensional data space (also called picture space) of Euclidean coordinates (e.g. pixels of an image) into a parameter space. The parameter space represents all possible 1D lines in the original 2D data space. In principle, each point of the data space is mapped into an infinite number of points in the parameter space which is not materialized as an infinite set but instead as a trigonometric function in the parameter space. Each function in the parameter space represents all lines in the picture space crossing the corresponding point in data space. The intersection of two curves in the parameter space indicates a line through both the corresponding points in the picture space.

The objective of a clustering algorithm is to find intersections of many curves in the parameter space representing lines through many database objects. The key feature of the Hough transform is that the distance of the points in the original data space is not considered any more. Objects can be identified as associated to a common line even if they are far apart in the original feature space. As a consequence, the Hough transform is a promising candidate for developing a principle for subspace analysis that does not require the locality assumption and, thus, enables a global subspace clustering approach. CASH follows a grid-based approach to identify dense regions in the parameter space, successively attribute-wise dividing the space and counting the functions intersecting each of the resulting hyperboxes. In a depth-first search, most promising paths in the search tree are searched first. A hyperbox is divided along one axis if it contains enough functions to allow for dense child boxes in turn. If a dense subspace is found, the algorithm is applied on the data set accounted for by the corresponding hyperbox projected on the corresponding subspace. This recursive descent allows for finding lower dimensional subspace clusters and implicitly yields a hierarchy of arbitrarily oriented subspaces and their accommodated clusters. However, if there are no correlation clusters in the original data space and, hence, no dense regions in the parameter space (but still, the hyperboxes remain dense enough to qualify as promising candidates), the complete search space is enumerated resulting in a worst case time complexity exponential in d. Probably, some more sophisticated heuristic may make this promising idea more practical for really high dimensional data.

CASH is a contribution of the author and will be discussed in more detail in Part IV.

6.2.3 Other Approaches

Other basic principles that have been considered suitable for correlation analysis and for the search for arbitrarily oriented subspace clusters in the literature include the concepts of the *fractal dimension* and of *random sampling* methods.

Fractal Dimension

There are some attempts to use the concept of self-similarity (fractal dimension) to cluster data sets [21, 108, 58]. This would provide quite a different basis to grasp correlations in addition to PCA and, therefore, constitutes a rather promising approach but does assume the locality of patterns even by definition. Furthermore, the fractal dimension as a single property of a data (sub-)set does not yield information regarding the primary directions within a data distribution and, hence, seems less helpful for correlation clustering than PCA-based techniques. Among approaches based on these principles not any proposition seems mature enough to base a fully developed clustering procedure on it. So it may require some effort to define really effective approaches to correlation clustering based on the fractal dimension. Nevertheless, we look forward to interesting new proposals in this field.

Random Sampling

In the area of pattern recognition, a correlation clustering procedure based on random sampling has been proposed in [62]: an l-dimensional subspace as a cluster hyperplane (technically speaking, a linear manifold) is constructed by sampling $l + 1$ points. The sampling is repeated n times, where n is a threshold to ensure that with a certain probability at least the points obtained by one sample indeed belong to a common cluster. Deriving this kind of threshold, however, is based on a couple of simplifying assumptions such as a rough estimate of the number of clusters and the points of the data set being equally distributed among all existing clusters. Assigning the points to the corresponding subspace cluster allows to assess the discriminability allowed for by the current clustering. Starting with 1-dimensional subspaces, the algorithm proceeds in a bottom-up manner. These considerations have the nice property that, in a sense, a correlation cluster model is fitted during the process.

In [65], the authors proposed a similar idea, this time based on RANSAC [52], an established algorithm to find 1-dimensional lines in a data set by random sampling. The derived 1-dimensional clusters are then refined by adding or removing features.

Unfortunately, in both cases, the experimental evaluation does not allow to estimate the merits of this approach in terms of efficiency or effectiveness in comparison to other correlation clustering approaches. Only a comparison with ORCLUS is presented in [62] where the results are rather inconclusive. However, the authors present convincing formalizations of the problem probably allowing for further generalizations. In summary, also in this approach we eagerly anticipate further enhancements.

6.3 Summary

As for biclustering approaches, a thorough evaluation of the different approaches in this field is still owing, especially in comparison to biclustering algorithms. Most biclustering approaches do not rely on the locality assumption, which makes them quite successful in many applications albeit their models are rather restricted compared to the model of correlation clusters. The idea to mine for objects exhibiting common patterns of correlations is pursued by both, pattern-based and correlation clustering. While pattern-based approaches mine for the sole patterns of similar behavior of data objects irrespective of their distance in Euclidean space, correlation clustering approaches widen the field of possible patterns to more complex, positive and negative correlations of attributes. The drawback of most correlation clustering approaches is their assumption that points of a cluster are still densely arranged in Euclidean space. Recent advances in correlation clustering, however, start to overcome the drawbacks of density-based approaches in high dimensional data in principle.

Chapter 7

Discussion

7.1 A Heuristic-Based Systematic View

The general approach to clustering in high dimensional data seeks to find clusters in arbitrarily oriented subspaces of the data space. So the general objective is: "*Find a partitioning of the data where each cluster may exist in its own subspace.*" The partitioning needs not to be unique, i.e., clusters may overlap. The subspaces may be axis-parallel or arbitrarily oriented and may or may not overlap.

Since clustering in general is an NP-hard problem, efficient solutions always use heuristics to yield approximate solutions in efficient time and space requirements. These heuristics are reflected usually in different cluster models and different algorithmic approaches like locally optimizing algorithms, greedy search procedures, etc. The problem of finding clusters in subspaces is in fact an even more complex task. A general, naïve solution would examine all possible subspaces to look for clusters. Clearly, this is impossible in the general case, since the search space of all possible arbitrarily oriented subspaces is infinite. Thus, assumptions and heuristics are required to make feasible solutions possible.

Such assumptions and heuristics are:

- the restriction of the search space to certain subspaces (e.g., axis-parallel subspaces) or certain patterns in the data matrix
- a clustering criterion implementing the downward-closure property
- selecting one axis of the data matrix as "dimensions", the other one as "data objects"
- the locality assumption
- the assumption of simple additive models ("patterns")
- randomized, greedy procedures
- specifying the number of clusters in advance

Any of the proposed methods is based on at least one assumption. Some corresponding properties of the most prominent algorithms are indicated in Table 7.1.

Subspace or projected clustering restricts the search space to axis-parallel subspaces and makes use of a clustering criterion implementing the downward closure property (usually based on a global density threshold) or makes use of the locality assumption to enable efficient search heuristics. For example, we find most bottom-up approaches here (CLIQUE, ENCLUS, MAFIA, SUBCLU, and P3C) free from the locality assumption. Instead, they pursue a complete enumeration approach facilitated by an APRIORI like search. Thus, they remain in $O(2^d)$ in the worst case.

Biclustering and pattern-based clustering approaches restrict the search space to special forms or locations of subspaces or half-spaces. If they include correlations among attributes (like δ-bicluster, FLOC, p-Cluster, MaPle, and CoClus), although they are free of the locality assumption, they are restricted to very special cases of correlations and pursue either a complete enumeration or require specification of the number of clusters in advance and perform a greedy search.

While biclustering approaches often treat rows and columns interchangeably, axis-parallel subspace and projected clustering as well as correlation

Table 7.1: Properties of clustering algorithms.

Algorithm	complex correlations	simple positive correlation	simple negative correlation	axis parallel	not relying on locality assumption	adaptive density threshold	independent w.r.t. order of attributes	independent w.r.t. order of objects	deterministic	arbitrary number of clusters	overlapping clusters	overlapping subspaces	simultaneously overlapping clusters and subspaces	arbitrary subspace dimensionality	hierarchical structure	avoiding complete enumeration	noise robust
axis parallel clustering																	
CLIQUE [13]				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
ENCLUS [40]				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
MAFIA [105]				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
SUBCLU [80]				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓			✓
PROCLUS [10]				✓		✓						✓				✓	
PreDeCon [34]				✓			✓	✓	✓	✓		✓				✓	✓
P3C [102]				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
COSA [53]				✓			✓	✓	✓			✓		✓		✓	✓
DOC [114]				✓	✓		✓	✓		✓	✓	✓	✓	✓			
DiSH [3]				✓	✓	✓		✓	✓	✓		✓		✓	✓	✓	✓
FIRES [88]				✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓
pattern-based clustering																	
Block clustering [66]					✓	*n a*	✓	✓	✓					✓	✓		✓
δ-bicluster [41]		✓	✓	✓	✓	*n a*	✓	✓	✓		✓	✓		✓		✓	✓
FLOC [144]		✓		✓	✓	*n a*					✓	✓	✓	✓		✓	✓
p-Cluster [138]		✓		✓	✓	*n a*	✓	✓	✓	✓	✓	✓	✓	✓			✓
MaPle [110]		✓		✓	✓	*n a*	✓	✓	✓	✓	✓	✓	✓	✓			✓
CoClus [42]		✓		✓	✓	*n a*								✓		✓	
OP-Cluster [96]					✓	*n a*	✓	✓	✓	✓	✓	*n a*	*n a*	*n a*			✓
correlation clustering																	
ORCLUS [11]	✓	✓	✓	✓			✓					✓				✓	
4C [35]	✓	✓	✓	✓			✓	✓	✓	✓		✓				✓	✓
COPAC [6]	✓	✓	✓	✓			✓	✓	✓	✓		✓		✓		✓	✓
ERiC [5]	✓	✓	✓	✓			✓	✓	✓	✓		✓		✓	✓	✓	✓
CASH [1]	✓	✓	✓	✓	✓	*n a*		✓	✓	✓		✓		✓	✓		✓

Heuristics restricting the general problem (see Section 7.1) are here formulated negatively, i.e., the more marks an algorithm features, the less assumptions are restricting the general search space (and, roughly, the more general is the algorithm).
Approaches neither marked to find positive or negative correlations, nor axis parallel clusters, are specialized to certain patterns that constitute very particular subspaces.
n a: not applicable

clustering approaches break this symmetry in favor of more efficient search heuristics or based on and motivated by specific spatial intuitions. This asymmetry between columns and rows may also be related to the point of view of a database researcher where columns correspond to attributes of a database entry and rows correspond to single database objects. Thus, in the context of a database, both types of information are addressed in a different way and it is assumed that the number of database entries (rows) by far exceeds the number of attributes (columns).

Almost all correlation clustering approaches proposed so far make use of the locality assumption but avoid complete enumeration. Hence they are efficient but their effectiveness may be questionable. An approach not relying on the locality assumption is CASH [1]. In turn, CASH suffers from the complete enumeration problem and remains rather inefficient. Another general way to get rid of the locality assumption is to base the clustering on a random sampling process. For these approaches, however, the quality of a retrieved clustering is a matter of luck.

7.2 A Problem-Oriented Systematic View

In Section 7.1, we took a view on the different algorithms considering their different assumptions and heuristics to restrict the most general search space. Another possible point of view is based on the specific problems associated with high dimensional data, commonly addressed all at once as the "curse of dimensionality".

7.2.1 "The Curse of Dimensionality" in the Clustering Problem

The term "curse of dimensionality" refers to a bundle of problems related to high dimensional data spaces. In the following, we list those problems that are most relevant for clustering high dimensional data.

Problem 1: Bellman is often cited for the term “curse of dimensionality” which he describes as “a malediction that has plagued the scientists from earliest days” [24, p. 94]. However, Bellman merely describes the fact that more dimensions result in more possibilities of values and disable finally a complete enumeration approach simply because tabularization and visualization of functions becomes increasingly difficult or even impossible with more variables. This problem is mainly known in pattern recognition and more elaborated in recent textbooks like [32].

Of course, this problem relates to the clustering problem in general: Seeking a clustering of a data set supposes the data being generated by several functions. Ideally, a clustering model would enable the user to identify the functional dependencies resulting in the data set at hand and, thus, to eventually find new and interesting insights in the laws of nature or economy or society or whatever domain the data set describes. Those functions are the more complex the more attributes contribute to the actual relationships.

Problem 2: Concepts like proximity, distance, or neighborhood become less meaningful with increasing dimensionality of a data set [31, 69, 9]. Roughly, the results in these papers state that the relative distance of the farthest point and the nearest point converges to 0 for increasing dimensionality d:

$$\lim_{d\to\infty} \frac{dist_{\max} - dist_{\min}}{dist_{\min}} \to 0,$$

i.e., discrimination between the nearest and the farthest neighbor becomes rather poor in high dimensional space. This is by far a more fundamental problem than the mere performance degradation of algorithms on high dimensional data.

Clearly, the “locality assumption” is somewhat naïve in view of this problem. As a solution, a more deliberate choice of distance metrics (e.g., the use of Manhattan distance or even fractional distance metrics) has been proposed [9]. However, this problem could be more fundamentally treated by dismissing any use of a local neighborhood in the clustering process.

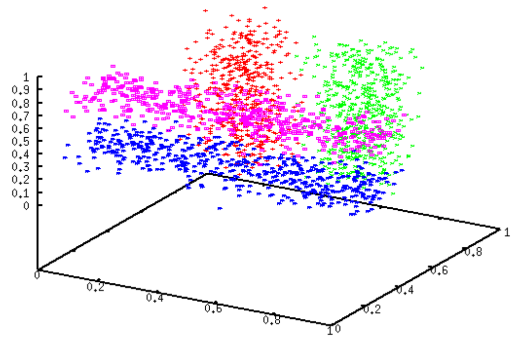
It is important to note that these observations are valid for a broad range of data distributions and occur simply based on the mere number of dimensions. This problem is independent of the following problem albeit the effect will be worsened considering Problem 3.

Problem 3: In order to find dependencies and laws describing some occurring phenomena a glut of data is collected and single entities are described with many possibly related attributes. Among those features, many irrelevant attributes can be expected. The relevance of certain attributes may differ for different groups of objects within the same data set. Thus, since groups of data are defined by some of the available attributes only, many irrelevant attributes may interfere with the efforts to find these groups. Irrelevant attributes can also be related to as “noise”. However, global feature reduction methods may be inadequate if there is no global noise but given sets of attributes are noisy only w.r.t. certain sets of objects.

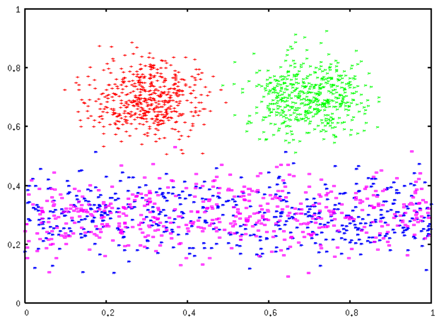
The challenge for clustering is therefore, related to this problem, to find an appropriate subset of attributes to describe the similarity of objects belonging to the same group and possibly different subsets of attributes for different groups of objects. The cluster objects, then, reside in axis-parallel, affine subspaces of the complete data space.

Although one could expect, as a rule of thumb, the more irrelevant features in a data set, the more dimensions there are at all, this problem can occur even in rather low dimensional data sets as it can be seen in Figure 7.1: four clusters in a 3-D data set are characterized by two relevant attributes, each generated by a Gaussian distribution. The values in the remaining attribute a uniformly distributed in $[0, 1]$ (cf. Figure 7.1(a)). For two clusters, however, the relevant attributes are x and y (Figure 7.1(b)) while for the remaining clusters, the relevant attributes are y and z (Figure 7.1(d)). Thus, these clusters can be discerned in the corresponding projections, the remaining clusters are intermixed. The third projection does also not allow a clear separation (Figure 7.1(c)). This problem is elaborated in more detail in [109].

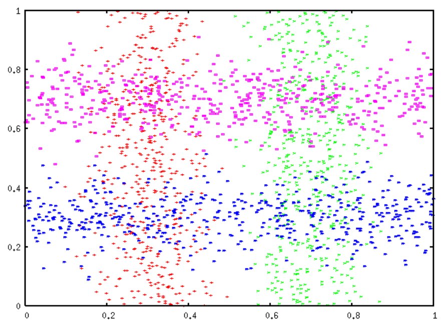
Many publications seem to obfuscate problems 2 and 3 but these are different effects in nature. However, irrespective of Problem 2, distance mea-



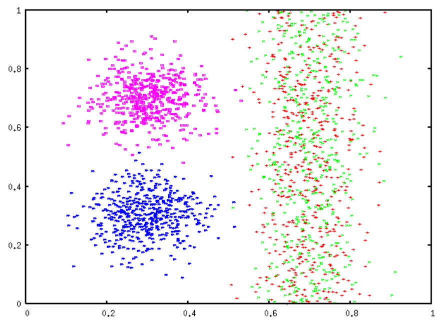
(a) Original data set.



(b) Projection on the subspace $\{x, y\}$



(c) Projection on the subspace $\{x, z\}$



(d) Projection on the subspace $\{y, z\}$

Figure 7.1: A 3-D data set illustrating Problem 3.

sures may be seriously misguided by irrelevant attributes.

Problem 4: Similarly as with Problem 3, in a data set containing many attributes, there may be some correlations among subsets of attributes. In sight of feature reduction methods, all but one of these attributes may be redundant. However, from the point of view of a domain scientist who collected these attributes in the first place, it may be an interesting new insight that there are so far unknown connections between features.

In view of spatial queries, the observation that the intrinsic dimensionality of a data set is often lower than the embedding dimensionality (based on interdependencies among attributes) is often seen as a solution to overcome the "curse of dimensionality" [48, 25, 106, 86]. In view of the clustering problem, however, Problems 1-3 remain unaffected by this phenomenon. In contrast, finding the correct subspace to define a suitable group of objects becomes a problem even harder since cluster objects may reside in arbitrarily oriented, affine subspaces (due to Problem 4).

Other Problems: There are quite some other problems related to the "curse of dimensionality". Most of those are, however, more relevant for indexing than for clustering data. Of course, clustering is affected by unbalanced range-queries largely searching in ranges beyond the boundaries of the data set and similar problems but these problems are elaborated extensively and in more detail in the literature concerned with index-structures (e.g. cf. [93, 30, 81, 28, 29, 140, 36, 37, 27]). So we base our attempt of a systematic view on subspace clustering approaches on Problems 1-4 as stated above.

7.2.2 Approaches as Solutions to Specific Problems

In view of the "curse of dimensionality", the special cases described in Section 7.1 above also reflect special problems.

While Problem 1 plagues all approaches in general, the remaining problems are differently tackled individually or in combination by different approaches. A possible remedy for Problem 2 is to avoid neighborhood queries. Thus, all approaches not relying on the locality assumption may be relatively unaffected by the problem of meaningless distance comparisons. Such approaches are especially bottom-up or hybrid approaches to axis-parallel subspace clustering and biclustering approaches (cf. Table 7.1). Note, however, that Problem 2, as opposed to Problems 3 and 4, occurs necessarily in rather high dimensional data and can occur in moderate dimensional data (i.e., $d = 10-15$). Problems 3 and 4, contrariwise, can occur in 2-dimensional data sets already. It becomes just more likely that these problems occur by chance with increasing data dimensionality. It is therefore important not to mistake Problem 2 for Problem 3 or vice versa.

Problem 3 leads to axis-parallel subspace clusters. So, all axis-parallel subspace and projected clustering approaches tackle especially this problem. Besides, this problem further worsens Problem 2.

Biclustering approaches tackle special forms of Problem 4: simple positive correlations between all attributes in a subset of the attributes. Some take further assumptions into consideration. The strong point of these approaches is being unconcerned about Problem 2 since they generally do not take any distances between database objects into account.

Correlation clustering specifically tackles Problem 4 in a general way. The occurrence of correlations among attributes alleviates Problems 2 and 3 in a way. However, with increasing dimensionality one cannot help considering those problems nevertheless.

The combination of Problems 3 and 4 will result in clusters in very sparse, arbitrarily oriented subspaces. We expect the next generation of subspace clustering algorithms to tackle these problems in combination, simultaneously considering Problem 2. A first shot is presented in [1], however, here strikes Problem 1 since this approach is trapped by the exponential time complexity of the complete enumeration in the worst case.

7.3 On the Difficulties in Solving Undefined Tasks

Among the multitude of algorithms, the task to solve remains often vague or undefined. It is often made clear which partial problems of the "curse of dimensionality" are tackled. But it remains unsaid what the meaning of the clusters retrieved by an algorithm exactly is. This vagueness is also an issue for comparatively evaluating different algorithms, as we will see later (Section 7.4). For now, we will concentrate on the obfuscation in discussing axis-parallel subspace and projected clustering algorithms resulting from confusing the two related but obviously not identical points of view for a classification of approaches: categorization according to the definition of the task vs. categorization according to algorithmic aspects. As we will see, it is part of the problem that both points of view are indeed closely interrelated.

7.3.1 Categorization w.r.t. Task-Definition

The problem definition of projected clustering algorithms – to find a unique partitioning of points into clusters – is obviously heavily influenced by the traditional full dimensional clustering problem. The only difference is that clusters may now exist in different subspaces of the original data which makes the problem much harder to solve. However, in the context of many applications, it is sensible that points may be assigned to different clusters in different subspaces. Thus, not allowing any overlap of the clusters may be too strong a limitation in these applications.

On the other hand, finding *all* clusters in *all* subspaces is a rather arbitrary problem definition due to two reasons. First, the number of clusters that are reported is usually very large. This overwhelmingly large set of clusters may quickly be to much for a user to analyze, interpret, and evaluate. Second, most of the clusters reported may be rather redundant because usually, any cluster in some k-dimensional subspace is also a cluster (or at least a subset of one) in all l-dimensional projections ($l < k$) of that subspace.

In fact, the subspace clustering problem can be seen as a justification of a bottom-up subspace search strategy that delivers exactly the desired solution. However, the relevance of this problem statement is at least questionable and heavily depends on the clustering criterion. Most bottom-up algorithms rely on a density-based cluster model and have to apply a global density threshold for all subspaces in order to meet the downward closure property. As a consequence, it is often not clear how meaningful the reported lower dimensional subspace clusters are.

Hybrid approaches seem to offer a good deal between the limitations of the problem statements of projected clustering and subspace clustering. They usually allow overlapping clusters but do not overwhelm the user with the (partly redundant) glut of all clusters in all subspaces. However, it is often not clear what each single hybrid algorithm exactly searches for.

7.3.2 Categorization w.r.t. Algorithmic Aspects

The problem of the *top-down* approaches is the circular dependency between cluster membership assignment and subspace learning from points of the cluster. In order to escape from this circular dependency, top-down approaches usually make the assumption that a subset of cluster members can be determined "somehow". In general, existing approaches implement two strategies for this "somehow": Many algorithms assume that the local neighborhood of cluster centers or other cluster members in the original full dimensional feature space contains a considerably large number of other cluster members (*locality assumption*). Other algorithms try to obtain cluster members from a random sample of points. The key point of these strategies is that the higher the number of outliers (non-cluster members) that are included in the selection, the less accurate will be the determination of the true subspace of the cluster. This has severe consequences because if the subspace of a cluster is not found correctly, again the assignment of points to that cluster may be less accurate. In sight of these considerations, both approaches – the locality assumption as well as random sampling – are usually rather strict limitations to the quality and applicability of the corresponding algorithms. A strong benefit of most top-down-approaches is a good worst-case scalability. Usually, a complete enumeration of the exponential search space is avoided also in the worst-case. Typically, algorithms implementing a top-down subspace search scale at most quadratic w.r.t. the dimensionality d of the data space.

In order to apply an efficient *bottom-up* subspace search approach similar to frequent itemset mining, the cluster criterion must implement the downward closure property. Existing bottom-up approaches usually rely on a density-based cluster criterion. A limitation of most of these approaches is that the cluster criterion must use a fixed density threshold for all subspaces in order to implement the downward closure property. As a consequence, the same globally defined density threshold applies for subspaces of considerably different dimensionality, although a significant cluster in a higher dimensional subspace will most likely be less dense (in an absolute sense) than a significant cluster in a lower dimensional subspace. In order to find higher dimensional subspaces, the user has to define a less strict density threshold. This, however, would produce a lot of meaningless lower dimensional clusters. On the other hand, choosing a more strict density threshold, the reported lower dimensional clusters will probably be more meaningful but higher dimensional subspace clusters will most likely be lost. In addition, bottom-up subspace search is a complete enumeration approach, i.e. the worst-case complexity is $O(2^d)$. On the average, usually the bottom-up search is also considerably less efficient than the top-down approach because in order to find a k-dimensional subspace, many 1-dimensional, 2-dimensional, $\ldots$, and $(k-1)$-dimensional subspaces need to be tested. As mentioned above, the fact that bottom-up approaches produce all lower dimensional projections of subspaces accommodating clusters during the bottom-up traversal of the search space can be seen as an advantage. Thus, for addressing the subspace clustering problem, a bottom-up strategy would generally make sense as far as the problem of meaningless lower dimensional subspace clusters is concisely addressed.

7.3.3 Summary

In summary, a big problem in the field of finding clusters in axis-parallel subspaces is that existing papers usually lack a meaningful task definition. Very often, the problem statement is geared to the proposed algorithm, i.e. the task that is to be solved is defined such that it matches the outcome of the proposed algorithm. For example, the algorithm PreDeCon computes a DBSCAN-like partitioning of the data but each cluster may exist in a different subspace. The problem statement of projected clustering perfectly matches this result. As a consequence, the accuracy and applicability of methods for finding clusters in axis-parallel subspaces is hard to compare because each algorithm produces different clusters not only because of different problem definitions but also because of the application of different cluster models.

Obviously, the problem statement should dictate the algorithm's outcome and not vice versa. But as long as a meaningful general problem statement is missing, no algorithms can be designed to tackle it. By then, it is likely that further algorithms with slightly different problem statements will be proposed. However, an important contribution to the field would be to carefully analyze relevant applications and to extract a meaningful problem definition that can be tackled.

7.4 Empirical Evaluation: A Desideratum

Newly proposed algorithms are often evaluated in a sloppy way taking into account only one or two competitors – if at all – or even with a so called "naïve" *ad hoc* solution for comparison of efficiency and effectiveness. Recently, an understanding for the need for consolidation of a maturing research area is rising in the research community as illustrated by the discussions about the repeatability of results for SIGMOD 2008, the Panel on performance evaluation at VLDB 2007, and the tentative special topic of "Experiments and Analyses Papers" at VLDB 2008. However, a fair and conclusive experimental evaluation of algorithms based on such a variety of assumptions and intuitions, pursuing so different search heuristics, and providing such a diversity of models and representations of results seems rather difficult.

Aside from comparing apples and oranges in an evaluation of different algorithms – what would qualify as a good experimental evaluation? By all means, only stating that clusters can be found by a given approach is far too less because any partitioning algorithm like k-means reports always clusters. A solid evaluation needs to analyze the clusters in order to show that the grouping reflects some domain specific knowledge. This is somehow frustrating because clustering as an unsupervised learning task aims at finding (beside already known information) previously unknown knowledge. However, the usefulness of newly derived knowledge can usually only be interpreted by a domain expert which is most likely not at hand. Thus, showing that a clustering method can reproduce existing knowledge and does not produce implications that contradict existing knowledge will be a rather solid statement. On the other hand, if a (true) domain expert can be asked to interpret the results, this would be the ne plus ultra in reliability of the experiments. Unfortunately, the latter is most likely the rarest though most expedient and, thus, thrilling scenario. After all, this is, why we should do data mining at all.

So far, there exists no complete competitive empirical evaluation w.r.t. efficiency and effectiveness of all or at least of the most prominent approaches. We sincerely hope that the systematic view and the theoretical comparison of different cluster models and objectives provided in this survey may be helpful for empirical studies in the future. However, a fair empirical evaluation of the different approaches is not a trivial task. The different heuristics and assumptions (cf. Section 7.1) and the different problems tackled (cf. Section 7.2) should always be kept in sight. In most cases, whether the trade-off between efficiency and effectiveness is tolerable will depend on the application.

Part III

Density-based Correlation Clustering

In this part, several contributions of the author to the field of correlation clustering as surveyed in Part II (see Chapter 6) are described in more detail. We focus in the following on density-based subspace clustering in arbitrarily oriented subspaces and describe the adaptation of the density-based clustering paradigm to the problem of correlation clustering.

The first, direct adaptation of density-based clustering to correlation clustering is presented in Chapter 8. This first approach has been enhanced in several ways w.r.t. efficiency as well as to effectiveness. These enhancements are described in Chapter 9.

Chapter 10 describes the first approach to mining hierarchies of correlation clusters. Some drawbacks of this first approach are described in Chapter 11, together with corresponding enhancements.

In Chapter 12, some weak points common to all approaches to correlation clustering based on PCA are discussed. Improvements within the local approach to correlation clustering, common to the density-based approaches discussed in this part, are proposed and evaluated.

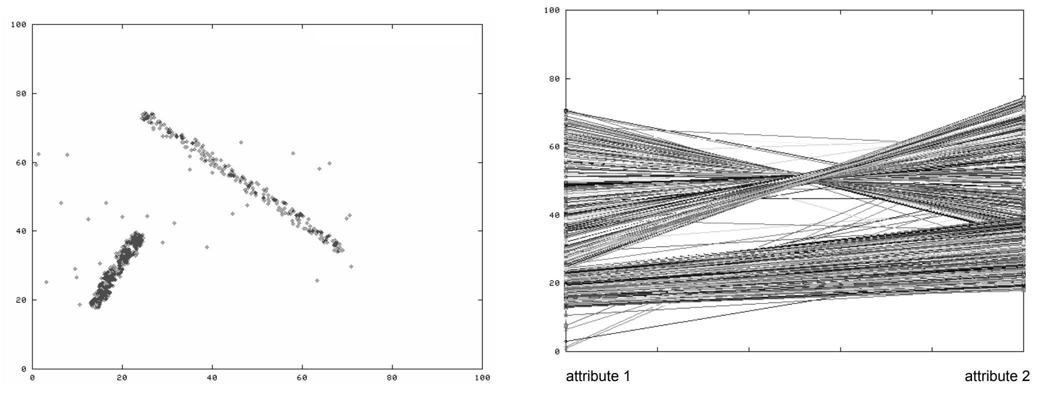
The single chapters in this part allow for a self-contained reading. This means, that some definitions may reoccur in several chapters in a similar way. As a result, to technically comprehend a given chapter reading the previous chapter is not required. In short, the technical contributions have been sketched in the survey of Chapter 6. Nevertheless, the sequence of chapters reflects a climax from solutions for simple to complex problems, all based on the density-based paradigm.

Chapter 8

Adapting the Density-based Paradigm for Correlation Clustering: 4C

In this chapter, we describe the first adaptation of the density-based clustering paradigm to the correlation clustering problem. Density-based clustering in general aims at partitioning the objects (described by points in a high dimensional feature space) of a data set into dense regions (clusters) separated by regions with low density (noise). Knowing the cluster structure is important and valuable because the different clusters often represent different classes of objects which have previously been unknown. Therefore, the clusters bring additional insight about the stored data set which can be exploited for various purposes such as improved marketing by customer segmentation, determination of homogeneous groups of web users through clustering of web logs, structuring large amounts of text documents by hierarchical clustering, or developing thematic maps from satellite images.

An interesting second kind of hidden information that may be interesting to users are correlations in a data set. A correlation is a linear dependency between two or more features (attributes) of the data set. The most important method for detecting correlations is the principal components analysis



(a) 2D view. (b) Parallel coordinate plot.

Figure 8.1: 1-Dimensional Correlation Lines

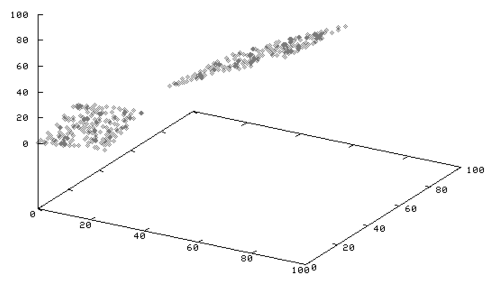
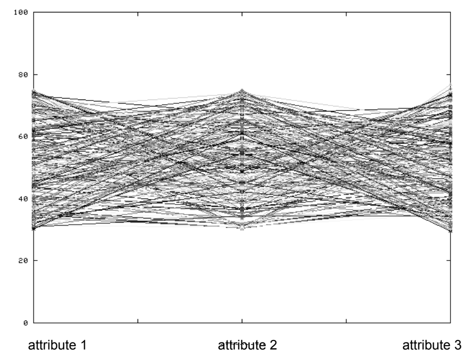
(PCA). Knowing correlations is also important and valuable because the dimensionality of the data set can be considerably reduced which improves both the performance of similarity search and data mining as well as the accuracy. Moreover, knowing about the existence of a relationship between attributes enables one to detect hidden causalities (e.g. the influence of the age of a patient and the dose rate of medication on the course of his disease or the co-regulation of gene expression) or to gain financial advantage (e.g. in stock quota analysis), etc.

Methods such as PCA, however, are restricted, because they can only be applied to the data set as a whole. Therefore, it is only possible to detect correlations which are expressed in all points or almost all points of the data set. For a lot of applications this is not the case. For instance, in the analysis of gene expression, we are facing the problem that a dependency between two genes does only exist under certain conditions. Therefore, the correlation is visible only in a local subset of the data. Other subsets may be either not correlated at all, or they may exhibit completely different kinds of correlation (different features are dependent on each other). The correlation of the whole data set can be weak, even if for local subsets of the data strong correlations exist. Figure 8.1 shows a simple example, where two subsets of 2-dimensional points exhibit different correlations.

To the best of our knowledge, both concepts of density-based clustering (i.e. finding densely populated subsets of the data) and correlation analysis have not yet been addressed as a combined task for data mining. The most relevant related approach is ORCLUS [11], but since it is k-medoid-based, it is very sensitive to noise and the locality of the analyzed correlations is usually too coarse, i.e., the number of objects taken into account for correlation analysis is too large. In this chapter, we present a new method which is capable of detecting local subsets of the data which exhibit strong correlations and which are densely populated (w.r.t. a given density threshold). We call such a subset a *correlation connected cluster*.

In lots of applications such correlation connected clusters are interesting. For example in E-commerce (recommendation systems or target marketing) where sets of customers with similar behavior need to be detected, one searches for positive linear correlations. In DNA microarray analysis (gene expression analysis) negative linear correlations express the fact that two genes may be co-regulated, i.e. if one has a high expression level, the other one is very low and *vice versa*. Usually such a co-regulation will only exist in a small subset of conditions or cases, i.e. the correlation will be hidden locally in the data set and cannot be detected by global techniques. Figures 8.1 and 8.2 show simple examples how correlation connected clusters can look like. In Figure 8.1 the attributes exhibit two different forms of linear correlation. We observe that if for some points there is a linear correlation of all attributes, these points are located along a line. Figure 8.2 presents two examples where an attribute z is correlated to attributes x and y (i.e., $z = a + bx + cy$). In this case the set of points forms a 2-dimensional plane.

As stated above, in this chapter, we propose an approach that meets both the goal of clustering and correlation analysis in order to find correlation connected clusters. The remainder is organized as follows: In Section 8.1 we formalize our notion of correlation connected clusters. Based on this formalization, we present in Section 8.2.1 an algorithm called 4C (*C*omputing *C*orrelation *C*onnected *C*lusters) to efficiently compute such correlation connected clusters and discuss the computational complexity and the parametrization of our algorithm, while Section 8.3 contains an extensive

(a) 3D view. (b) Parallel coordinate plot of one plane.

Figure 8.2: 2-Dimensional Correlation Planes

experimental evaluation of 4C.

Parts of the material presented in this chapter has been published in a similar way in [35]. In comparison to this earlier publication, some minor errors and notational flaws are corrected.

8.1 The Notion of Correlation Connected Clusters

In this section, we formalize the notion of a correlation connected cluster. Let $\mathcal{D}$ be a database of d-dimensional feature vectors ($\mathcal{D} \subseteq \mathbb{R}^d$). An element $P \in \mathcal{D}$ is called point or object. The value of the i-th attribute ($1 \leq i \leq d$) of P is denoted by p_i (i.e. $P = (p_1, \ldots, p_d)^\mathsf{T}$). Intuitively, a correlation connected cluster is a dense region of points in the d-dimensional feature space having at least one principal axis with low variation along this axis. Thus, a correlation connected cluster has two different properties: density and correlation. In the following, we will first address these two properties and then merge these ingredients to formalize our notion of correlation connected clusters.

8.1.1 Density-Connected Sets

The density-based notion is a common approach for clustering used by various clustering algorithms such as DBSCAN [47], DBCLASD [143], DENCLUE [70], and OPTICS [16]. All these methods search for regions of high density in a feature space that are separated by regions of lower density.

A typical density-based clustering algorithm needs two parameters to define the notion of density: First, a parameter *MinPts* specifying the minimum number of objects, and second, a parameter ε specifying a volume. These two parameters determine a density threshold for clustering.

Our approach follows the formal definitions of density-based clusters underlying the algorithm DBSCAN. The formal definition of the clustering notion is presented and discussed in full details in [47]. In the following we give a short summary of all necessary definitions.

Definition 8.1 (ε-neighborhood)
Let $\varepsilon \in \mathbb{R}^+$ and $O \in \mathcal{D}$. The ε-neighborhood *of O, denoted by $\mathcal{N}_\varepsilon^O$, is defined by*

$$\mathcal{N}_\varepsilon^O = \{X \in \mathcal{D} \mid \mathit{dist}(O, X) \leq \varepsilon\}.$$

Based on the two input parameters ε and *MinPts*, dense regions can be defined by means of core objects:

Definition 8.2 (core object)
Let $\varepsilon \in \mathbb{R}^+$ and $\mathit{MinPts} \in \mathbb{N}$. An object $O \in \mathcal{D}$ is called core object *w.r.t. ε and MinPts, if its ε-neighborhood contains at least MinPts objects, formally:*

$$\textsc{Core}_{\mathit{den}}(O) \Leftrightarrow |\mathcal{N}_\varepsilon^O| \geq \mathit{MinPts}.$$

Let us note, that we use the acronym "den" for the density parameters ε and *MinPts*. In the following, we omit the parameters ε and *MinPts* wherever the context is clear and use "den" instead.

A core object O can be used to expand a cluster, with all the density-connected objects of O. To find these objects the following concepts are used.

Definition 8.3 (direct density-reachability)
Let $\varepsilon \in \mathbb{R}^+$ and MinPts $\in \mathbb{N}$. An object $P \in \mathcal{D}$ is directly density-reachable *from $Q \in \mathcal{D}$ w.r.t. ε and MinPts, if Q is a core object and P is an element of $\mathcal{N}_\varepsilon^Q$, formally:*

$$\textsc{DirReach}_{den}(Q,P) \Leftrightarrow \textsc{Core}_{den}(Q) \ \wedge \ P \in \mathcal{N}_\varepsilon^Q.$$

Let us note, that direct density-reachability is symmetric only for core objects.

Definition 8.4 (density-reachability)
Let $\varepsilon \in \mathbb{R}^+$ and MinPts $\in \mathbb{N}$. An object $P \in \mathcal{D}$ is density-reachable *from $Q \in \mathcal{D}$ w.r.t. ε and MinPts, if there is a chain of objects $P_1, \ldots, P_n \in \mathcal{D}$, $P_1 = Q$, $P_n = P$ such that P_{i+1} is directly density-reachable from P_i, formally:*

$$\begin{aligned}&\textsc{Reach}_{den}(Q,P) \Leftrightarrow \\ &\quad \exists P_1, \ldots, P_n \in \mathcal{D} : P_1 = Q \ \wedge \ P_n = P \ \wedge \\ &\quad \forall i \in \{1, \ldots, n-1\} : \textsc{DirReach}_{den}(P_i, P_{i+1}).\end{aligned}$$

Density-reachability is the transitive closure of direct density-reachability. However, it is still not symmetric in general.

Definition 8.5 (density-connectivity)
Let $\varepsilon \in \mathbb{R}^+$ and MinPts $\in \mathbb{N}$. An object $P \in \mathcal{D}$ is density-connected *to an object $Q \in \mathcal{D}$ w.r.t. ε and MinPts, if there is an object $O \in \mathcal{D}$ such that both P and Q are density-reachable from O, formally:*

$$\begin{aligned}&\textsc{Connect}_{den}(Q,P) \Leftrightarrow \\ &\exists O \in \mathcal{D} : \textsc{Reach}_{den}(O,Q) \ \wedge \ \textsc{Reach}_{den}(O,P).\end{aligned}$$

Density-connectivity is a symmetric relation. A density-connected cluster is defined as a set of density-connected objects which is maximal w.r.t. density-reachability [47].

Definition 8.6 (density-connected set)
Let $\varepsilon \in \mathbb{R}^+$ and $MinPts \in \mathbb{N}$. A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ is called a density-connected set *w.r.t. ε and MinPts, if all objects in $\mathcal{C}$ are density-connected and $\mathcal{C}$ is maximal w.r.t. density-reachability, formally:*

$\textsc{ConSet}_{den}(\mathcal{C}) \Leftrightarrow$

(1) *Connectivity:* $\forall O, Q \in \mathcal{C} : \textsc{Connect}_{den}(O, Q)$

(2) *Maximality:* $\forall P, Q \in \mathcal{D} : Q \in \mathcal{C} \wedge \textsc{Reach}_{den}(Q, P) \Rightarrow P \in \mathcal{C}$.

Using these concepts 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-objects O and computing all objects which are density reachable from O. The correctness of DBSCAN can be formally proven (cf. Lemmata 1 and 2 in [47], proofs in [120]). Although DBSCAN is not in a strong sense deterministic (the run of the algorithm depends on the order in which the points are stored), both the run-time as well as the result (number of detected clusters and association of core objects to clusters) are determinate. The worst case time complexity of DBSCAN is $O(n \log n)$ assuming an efficient index and $O(n^2)$ if no index exists.

8.1.2 Correlation Sets

In order to identify correlation connected clusters (regions in which the points exhibit correlation) and to distinguish them from usual clusters (regions of high point density only) we are interested in all sets of points with an intrinsic dimensionality that is considerably smaller than the embedding dimensionality of the data space (e.g. a line or a plane in a three or higher dimensional space). There are several methods to measure the intrinsic dimensionality of a point set in a region, such as the fractal dimension or the principal components analysis (PCA). We choose PCA because the fractal dimension appeared to be not stable enough in our first experiments.

The PCA determines the covariance matrix $\boldsymbol{M} = [m_{ij}]$ with $m_{ij} = \sum\limits_{S \in \mathcal{S}} (s_i - \bar{s}_i) \cdot (s_j - \bar{s}_j)$ of the considered point set $\mathcal{S}$ where $\bar{s}_i$ is the mean of all points $S \in \mathcal{S}$ in attribute i, and decomposes it into an orthonormal matrix $\boldsymbol{V}$ called eigenvector matrix and a diagonal matrix $\boldsymbol{E}$ called eigenvalue matrix such that $\boldsymbol{M} = \boldsymbol{V} \cdot \boldsymbol{E} \cdot \boldsymbol{V}^{\mathsf{T}}$. The eigenvectors represent the principal axes of the data set (as normalized by linear translation to the origin) whereas the eigenvalues represent the variance along these axes. In case of a linear dependency between two or more attributes of the point set (correlation), one or more eigenvalues are close to zero.

A set forms a λ-dimensional correlation hyperplane if $d - \lambda$ eigenvalues fall below a given threshold $\delta \approx 0$. Since the eigenvalues of different sets exhibiting different densities may differ a lot in their absolute values, we normalize the eigenvalues by mapping them onto the interval $[0, 1]$. This normalization is denoted by Ω and simply divides each eigenvalue e_i by the maximum eigenvalue e_{max}. We call the eigenvalues e_i with $\Omega(e_i) \leq \delta$ *close to zero.*

Definition 8.7 (λ-dimensional linear correlation set)
Let $\mathcal{S} \subseteq \mathcal{D}$, $\lambda \in \mathbb{N}$ ($\lambda \leq d$), $\boldsymbol{E} = e_1, ..., e_d$ the eigenvalues of $\mathcal{S}$ in descending order (i.e. $e_i \geq e_{i+1}$) and $\delta \in \mathbb{R}^+$ ($\delta \approx 0$). $\mathcal{S}$ forms an λ-dimensional linear correlation set *w.r.t. δ if at least $d - \lambda$ eigenvalues of $\mathcal{S}$ are close to zero, formally:*

$$\textsc{CorSet}^{\lambda}_{\delta}(\mathcal{S}) \Leftrightarrow |\{e_i \in \boldsymbol{E} \mid \Omega(e_i) \leq \delta\}| \geq d - \lambda.$$

where $\Omega(e_i) = e_i / e_1$.

This condition states that the variance of $\mathcal{S}$ along $d - \lambda$ principal axes is low and therefore the objects of $\mathcal{S}$ form a λ-dimensional hyperplane. We drop the index λ and speak of a correlation set in the following wherever it is clear from context.

Definition 8.8 (correlation dimension)
Let $\mathcal{S} \in \mathcal{D}$ be a linear correlation set w.r.t. $\delta \in \mathbb{R}^+$. The number of eigenvalues with $e_i > \delta$ is called correlation dimension *of $\mathcal{S}$, denoted by* $\textsc{CorDim}(\mathcal{S})$.

Let us note, that if $\mathcal{S}$ is a λ-dimensional linear correlation set, then $\textsc{CorDim}(\mathcal{S}) \leq \lambda$. The correlation dimension of a linear correlation set $\mathcal{S}$ corresponds to the intrinsic dimension of $\mathcal{S}$.

8.1.3 Clusters as Correlation-Connected Sets

A correlation connected cluster can be regarded as a maximal set of density-connected points that exhibit uniform correlation. We can formalize the concept of correlation connected sets by merging the concepts described in the previous two subsections: density-connected sets (cf. Definition 8.6) and correlation sets (cf. Definition 8.7). The intuition of our formalization is to consider those points as core objects of a cluster which have an appropriate correlation dimension in their neighborhood. Therefore, we associate each point P with a similarity matrix $\boldsymbol{M}_P$ which is determined by PCA of the points in the ε-neighborhood of P. For convenience we call $\boldsymbol{V}_P$ and $\boldsymbol{E}_P$ the eigenvectors and eigenvalues of P, respectively. A point P is inserted into a cluster if it has the same or a similar similarity matrix like the points in the cluster. To achieve this goal, our algorithm looks for points that are close to the principal axis (or axes) of those points which are already in the cluster. We will define a similarity measure $\hat{\boldsymbol{M}}_P$ for the efficient search of such points.

We start with the formal definition of the covariance matrix $\boldsymbol{M}_P$ associated with a point P.

Definition 8.9 (covariance matrix of a point)
Let $P \in \mathcal{D}$. The matrix $\boldsymbol{M}_P = [m_{ij}]$ with

$$m_{ij} = \sum_{S \in \mathcal{N}_\varepsilon^P} (s_i - \bar{s}_i) \cdot (s_j - \bar{s}_j) \qquad (1 \leq i, j \leq d),$$

where $\bar{s}_i$ is the mean of all points $S \in \mathcal{N}_\varepsilon^P$ in attribute i, is called the covariance matrix *of the point P. $\boldsymbol{V}_P$ and $\boldsymbol{E}_P$ (with $\boldsymbol{M}_P = \boldsymbol{V}_P \cdot \boldsymbol{E}_P \cdot \boldsymbol{V}_P^{\mathsf{T}}$) as*

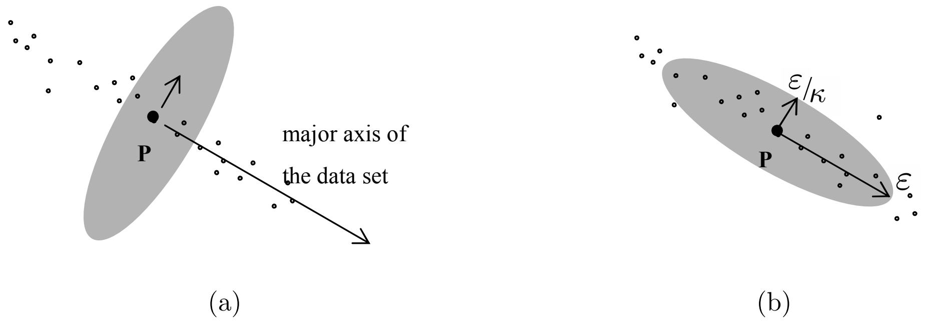


Figure 8.3: Correlation ε-neighborhood of a point P according to (a) $\boldsymbol{M}_P$ and (b) $\hat{\boldsymbol{M}}_P$.

determined by PCA of $\boldsymbol{M}_P$ are called the eigenvectors and eigenvalues of the point P, respectively.

We can now define the new similarity measure $\hat{\boldsymbol{M}}_P$ which searches points in the direction of highest variance of $\boldsymbol{M}_P$ (the major axes). Theoretically, $\boldsymbol{M}_P$ could be directly used as a similarity measure, i.e.

$$dist_{\boldsymbol{M}_P}(P,Q) = \sqrt{(P-Q)^{\mathsf{T}} \cdot \boldsymbol{M}_P \cdot (P-Q)} \quad \text{where } P,Q \in \mathcal{D}.$$

Figure 8.3(a) shows the set of points which lies in an ε-neighborhood of the point using $\boldsymbol{M}_P$ as similarity measure. The distance measure puts high weights on those axes with a high variance whereas directions with a low variance are associated with low weights. This is usually desired in similarity search applications where directions of high variance have a high distinguishing power and, in contrast, directions of low variance are negligible.

Obviously, for our purpose of detecting correlation clusters, we need quite the opposite. We want so search for points in the direction of highest variance of the data set. Therefore, we need to assign low weights to the direction of highest variance in order to shape the ellipsoid such that it reflects the data distribution (cf. Figure 8.3(b)). The solution is to change large eigenvalues into smaller ones and *vice versa*. We use two fixed values, 1 and a parameter $\kappa \gg 1$ rather than e.g. inverting the eigenvalues in order to avoid problems with singular covariance matrices. The number 1 is a natural choice because the corresponding semi-axes of the ellipsoid are then epsilon. The parameter κ controls the “thickness” of the λ-dimensional correlation line or plane, i.e. the tolerated deviation.

This is formally captured in the following definition:

Definition 8.10 (correlation similarity matrix of a point)
Let $P \in \mathcal{D}$ and let $\boldsymbol{V}_P$, $\boldsymbol{E}_P$ be the corresponding eigenvectors and eigenvalues of the point P. Let $\kappa \in \mathbb{R}$ be a constant with $\kappa \gg 1$. The new eigenvalue matrix $\hat{\boldsymbol{E}}_P$ with diagonal entries $\hat{e}_i$ $(i = 1, \dots d)$ is computed from the eigenvalues $e_1, \dots, e_d$ in $\boldsymbol{E}_P$ according to the following rule:

$$\hat{e}_i = \begin{cases} 1 & \text{if} \quad \Omega(e_i) > \delta \\ \kappa & \text{if} \quad \Omega(e_i) \leq \delta \end{cases}$$

where Ω is the normalization of the eigenvalues onto $[0, 1]$ *as described above. The matrix $\hat{\boldsymbol{M}}_P = \boldsymbol{V}_P \cdot \hat{\boldsymbol{E}}_P \cdot \boldsymbol{V}_P^{\mathsf{T}}$ is called the* correlation similarity matrix. *The* correlation similarity measure *associated with point P is denoted by*

$$cdist_P(P, Q) = \sqrt{(P - Q)^{\mathsf{T}} \cdot \hat{\boldsymbol{M}}_P \cdot (P - Q)}.$$

Figure 8.3(b) shows the ε-neighborhood according to the correlation similarity matrix $\hat{\boldsymbol{M}}_P$. As described above, the parameter κ specifies how much deviation from the correlation is allowed. The greater the parameter κ, the tighter and clearer the correlations which will be computed. It empirically turned out that our algorithm presented in Section 8.2.1 is rather insensitive to the choice of κ. A good suggestion is to set $\kappa = 50$ in order to achieve satisfying results, thus — for the sake of simplicity — we omit the parameter κ in the following.

Using this similarity measure, we can define the notions of correlation core objects and correlation reachability. However, in order to define correlation-connectivity as a symmetric relation, we face the problem that the similarity measure in Definition 8.10 is not symmetric, because $dist_P(P, Q) = dist_Q(Q, P)$ does in general not hold (cf. Figure 8.4(b)). Symmetry, however, is important to avoid ambiguity of the clustering result. If an asymmetric

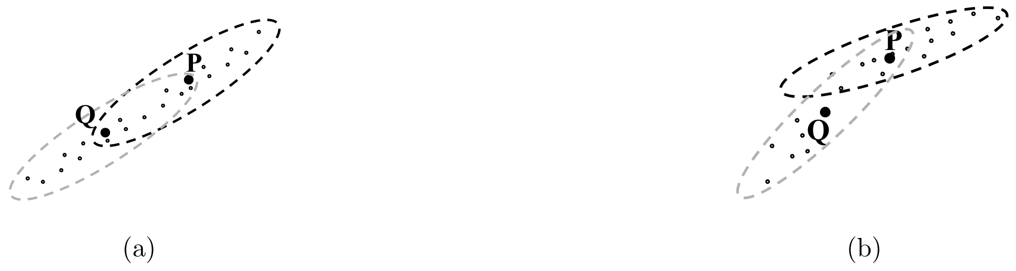


Figure 8.4: Symmetry of the correlation ε-neighborhood: (a) $P \in \mathcal{N}_\varepsilon^{\hat{\boldsymbol{M}}_Q}(Q)$. (b) $P \notin \mathcal{N}_\varepsilon^{\hat{\boldsymbol{M}}_Q}(Q)$.

similarity measure is used in DBSCAN a different clustering result can be obtained depending on the order of processing (e.g. which point is selected as the starting object) because the symmetry of density-connectivity depends on the symmetry of direct density-reachability for core-objects. Although the result is typically not seriously affected by this ambiguity effect we avoid this problem easily by an extension of our similarity measure which makes it symmetric. The trick is to consider both similarity measures, $dist_P(P,Q)$ as well as $dist_Q(P,Q)$ and to combine them by a suitable arithmetic operation such as the maximum of the two. Based on these considerations, we define the correlation ε-neighborhood as a symmetric concept:

Definition 8.11 (correlation ε-neighborhood)
Let $\varepsilon \in \mathbb{R}^+$. The correlation ε-neighborhood *of an object $O \in \mathcal{D}$, denoted by $\mathcal{N}_\varepsilon^{\hat{\boldsymbol{M}}_O}(O)$, is defined by:*

$$\mathcal{N}_\varepsilon^{\hat{\boldsymbol{M}}_O}(O) = \{X \in \mathcal{D} \mid \max\{cdist_O(O,X), cdist_X(X,O)\} \leq \varepsilon\}.$$

The symmetry of the correlation ε-neighborhood is illustrated in Figure 8.4. Correlation core objects can now be defined as follows.

Definition 8.12 (correlation core object)
Let $\varepsilon, \delta \in \mathbb{R}^+$ and $MinPts, \lambda \in \mathbb{N}$. A point $O \in D$ is called correlation core object *w.r.t. ε, $MinPts$, δ, and λ (denoted by $\text{CORE}_{den}^{cor}(O)$), if its ε-*

neighborhood is a λ-dimensional linear correlation set and its correlation ε-neighborhood contains at least MinPts points, formally:

$$\text{CORE}^{cor}_{den}(O) \Leftrightarrow \text{CORSET}^{\lambda}_{\delta}(\mathcal{N}^{P}_{\varepsilon}) \wedge |\,\mathcal{N}^{\hat{\boldsymbol{M}}_P}_{\varepsilon}(P)\,| \geq \mathit{MinPts}.$$

Let us note that in CORE^{cor}_{den} the acronym "cor" refers to the correlation parameters δ and λ. In the following, we omit the parameters ε, *MinPts*, δ, and λ wherever the context is clear and use "den" and "cor" instead.

Definition 8.13 (direct correlation-reachability)
Let $\varepsilon, \delta \in \mathbb{R}^+$ and $\mathit{MinPts}, \lambda \in \mathbb{N}$. A point $P \in \mathcal{D}$ is direct correlation-reachable *from a point $Q \in \mathcal{D}$ w.r.t. ε, MinPts, δ, and λ (denoted by* $\text{DIRREACH}^{cor}_{den}$*(Q,P)) if Q is a correlation core object, the correlation dimension of $\mathcal{N}^{P}_{\varepsilon}$ is at least λ, and $P \in \mathcal{N}^{\hat{\boldsymbol{M}}_Q}_{\varepsilon}(Q)$, formally:*

$\text{DIRREACH}^{cor}_{den}(Q,P) \Leftrightarrow$

(1) $\text{CORE}^{cor}_{den}(Q)$

(2) $\text{CORDIM}(\mathcal{N}^{P}_{\varepsilon}) \leq \lambda$

(3) $P \in \mathcal{N}^{\hat{\boldsymbol{M}}_Q}_{\varepsilon}(Q)$.

Correlation-reachability is symmetric for correlation core objects. Both objects P and Q must find the other object in their corresponding correlation ε-neighborhood.

Definition 8.14 (correlation-reachability)
Let $\varepsilon, \delta \in \mathbb{R}^+$ $(\delta \approx 0)$ and $\mathit{MinPts}, \lambda \in \mathbb{N}$. An object $P \in \mathcal{D}$ is correlation-reachable *from an object $Q \in \mathcal{D}$ w.r.t. ε, MinPts, δ, and λ (denoted by* REACH^{cor}_{den}*(Q,P)), if there is a chain of objects $P_1, \cdots, P_n$ such that $P_1 = Q, P_n = P$ and P_{i+1} is direct correlation-reachable from P_i, formally:*

$$\begin{aligned}&\text{REACH}^{cor}_{den}(Q,P) \Leftrightarrow\\ &\quad \exists P_1, \ldots, P_n \in \mathcal{D} : P_1 = Q \;\wedge\; P_n = P \;\wedge\\ &\quad \forall i \in \{1, \ldots, n-1\} : \text{DIRREACH}^{cor}_{den}(P_i, P_{i+1}).\end{aligned}$$

It is easy to see, that correlation-reachability is the transitive closure of direct correlation-reachability.

Definition 8.15 (correlation-connectivity)
Let $\varepsilon, \delta \in \mathbb{R}^+$ and $MinPts, \lambda \in \mathbb{N}$. An object $P \in \mathcal{D}$ is correlation-connected *to an object $Q \in \mathcal{D}$ if there is an object $O \in \mathcal{D}$ such that both P and Q are correlation-reachable from O, formally:*

$$\begin{aligned}&\textsc{Connect}_{den}^{corr}(Q,P) \Leftrightarrow\\ &\exists o \in \mathcal{D} \,:\, \textsc{Reach}_{den}^{corr}(O,Q) \;\wedge\; \textsc{Reach}_{den}^{corr}(O,P).\end{aligned}$$

Correlation-connectivity is a symmetric relation. A correlation-connected cluster can now be defined as a maximal correlation-connected set:

Definition 8.16 (correlation-connected set)
Let $\varepsilon, \delta \in \mathbb{R}^+$ and $MinPts, \lambda \in \mathbb{N}$. A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ is called a density-connected set *w.r.t. ε, MinPts, δ, and λ, if all objects in $\mathcal{C}$ are density-connected and $\mathcal{C}$ is maximal w.r.t. density-reachability, formally:*

$\textsc{ConSet}_{den}^{cor}(\mathcal{C}) \Leftrightarrow$

(1) *Connectivity:* $\forall O, Q \in \mathcal{C} : \textsc{Connect}_{den}^{cor}(O,Q)$

(2) *Maximality:* $\forall P, Q \in \mathcal{D} : Q \in \mathcal{C} \wedge \textsc{Reach}_{den}^{cor}(Q,P) \Rightarrow P \in \mathcal{C}$.

The following two lemmata are important for validating the correctness of our clustering algorithm. Intuitively, they state that we can discover a correlation-connected set for a given parameter setting in a two-step approach: First, choose an arbitrary correlation core object O from the database. Second, retrieve all objects that are correlation-reachable from O. This approach yields the density-connected set containing O.

Lemma 8.1
Let $P \in \mathcal{D}$. If P is a correlation core object, then the set of objects, which are correlation-reachable from P is a correlation-connected set, formally:

$$\begin{aligned}\textsc{Core}_{den}^{cor}(P) \wedge \mathcal{C} = \{O \in \mathcal{D} \,|\, \textsc{Reach}_{den}^{cor}(P,O)\}&\\ \Rightarrow \textsc{ConSet}_{den}^{cor}(\mathcal{C}).&\end{aligned}$$

Proof.

(1) $\mathcal{C} \neq \emptyset$*:*
By assumption, $\textsc{Core}_{den}^{cor}(P)$ *and thus,* $\textsc{CorDim}(\mathcal{N}_\varepsilon^P) \leq \lambda$.
$\Rightarrow \textsc{DirReach}_{den}^{cor}(P,P)$
$\Rightarrow \textsc{Reach}_{den}^{cor}(P,P)$
$\Rightarrow P \in \mathcal{C}$.

(2) Maximality:
Let $X \in \mathcal{C}$ *and* $Y \in \mathcal{D}$ *and* $\textsc{Reach}_{den}^{cor}(X,Y)$.
$\Rightarrow \textsc{Reach}_{den}^{cor}(P,X) \wedge \textsc{Reach}_{den}^{cor}(X,Y)$
$\Rightarrow \textsc{Reach}_{den}^{cor}(P,Y)$ *(since correlation reachability is a transitive relation).*
$\Rightarrow Y \in \mathcal{C}$.

(3) Connectivity:
$\forall X, Y \in \mathcal{C} : \textsc{Reach}_{den}^{cor}(P,X) \wedge \textsc{Reach}_{den}^{cor}(P,Y)$
$\Rightarrow \textsc{Connect}_{den}^{cor}(X,Y)$ *(via* P*).* $\square$

Lemma 8.2
Let $\mathcal{C} \subseteq \mathcal{D}$ *be a correlation-connected set. Let* $P \in \mathcal{C}$ *be a correlation core object. Then* $\mathcal{C}$ *equals the set of objects which are correlation-reachable from* P*, formally:*

$$\begin{aligned}&\textsc{ConSet}_{den}^{cor}(\mathcal{C}) \wedge P \in \mathcal{C} \wedge \textsc{Core}_{den}^{cor}(P)\\ &\quad \Rightarrow \mathcal{C} = \{O \in \mathcal{D} \,|\, \textsc{Reach}_{den}^{cor}(P,O)\}.\end{aligned}$$

Proof.
Let $\bar{\mathcal{C}} = \{O \in \mathcal{D} \,|\, \textsc{Reach}_{den}^{cor}(P,O)\}$. *We have to show that* $\bar{\mathcal{C}} = \mathcal{C}$*:*

(1) $\bar{\mathcal{C}} \subseteq \mathcal{C}$*: obvious from definition of* $\bar{\mathcal{C}}$.

(2) $\mathcal{C} \subseteq \bar{\mathcal{C}}$*: Let* $Q \in \mathcal{C}$*. By assumption,* $P \in \mathcal{C}$ *and* $\textsc{ConSet}_{den}^{cor}(\mathcal{C})$.
$\Rightarrow \exists O \in \mathcal{C} : \textsc{Reach}_{den}^{cor}(O,P) \wedge \textsc{Reach}_{den}^{cor}(O,Q)$
$\Rightarrow \textsc{Reach}_{den}^{cor}(P,O)$ *(since both* O *and* P *are correlation core objects and correlation reachability is symmetric for correlation core objects.*
$\Rightarrow \textsc{Reach}_{den}^{cor}(P,Q)$ *(transitivity of correlation-reachability)*
$\Rightarrow Q \in \bar{\mathcal{C}}$. $\square$

8.2 Computing Correlation Connected Clusters

8.2.1 Algorithm 4C

In the following we describe the algorithm 4C, which performs one pass over the database to find all correlation clusters for a given parameter setting. The pseudo code of the algorithm 4C is given in Figure 8.5. At the beginning each object is marked as unclassified. During the run of 4C all objects are either assigned a certain cluster identifier or marked as noise. For each object which is not yet classified, 4C checks whether this object is a correlation core object (see STEP 1 in Figure 8.5). If the object is a correlation core object the algorithm expands the cluster belonging to this object (STEP 2.1). Otherwise the object is marked as noise (STEP 2.2). To find a new cluster, 4C starts in STEP 2.1 with an arbitrary correlation core object O and searches for all objects that are correlation-reachable from O. This is sufficient to find the whole cluster containing the object O, due to Lemma 8.2. When 4C enters STEP 2.1 a new cluster identifier "clusterID" is generated which will be assigned to all objects found in STEP 2.1. 4C begins by inserting all objects in the correlation ε-neighborhood of object O into a queue. For each object in the queue it computes all directly correlation reachable objects and inserts those objects into the queue which are still unclassified. This is repeated until the queue is empty.

As discussed in Section 8.1 the results of 4C do not depend on the order of processing, i.e. the resulting clustering (number of clusters and association of core objects to clusters) is determinate.

8.2.2 Complexity Analysis

The computational complexity with respect to the number of data points as well as the dimensionality of the data space is an important issue because the proposed algorithms are typically applied to large data sets of high di-

```
algorithm 4C($\mathcal{D}$, $\varepsilon$, $MinPts$, $\lambda$, $\delta$)

  // assumption: each object in $\mathcal{D}$ is marked as unclassified

  for each unclassified $O \in \mathcal{D}$ do

STEP 1. test $\textsc{Core}^{cor}_{den}(O)$ predicate:

    compute $\mathcal{N}^{O}_{\varepsilon}$;
    if $|\mathcal{N}^{O}_{\varepsilon}| \geq MinPts$ then
      compute $\boldsymbol{M}_O$;
      if $\textsc{CorDim}(\mathcal{N}^{O}_{\varepsilon}) \leq \lambda$ then
        compute $\hat{\boldsymbol{M}}_O$ and $\mathcal{N}^{\hat{\boldsymbol{M}}_O}_{\varepsilon}(O)$;
        test $|\mathcal{N}^{\hat{\boldsymbol{M}}_O}_{\varepsilon}(O)| \geq MinPts$;

STEP 2.1. if $\textsc{Core}^{cor}_{den}(O)$ expand a new cluster:

    generate new clusterID;
    insert all $X \in \mathcal{N}^{\hat{\boldsymbol{M}}_O}_{\varepsilon}(O)$ into queue $\Phi$;
    while $\Phi \neq \emptyset$ do
      $Q =$ first object in $\Phi$;
      compute $\mathcal{R} = \{X \in \mathcal{D} \,|\, \textsc{DirReach}^{cor}_{den}(Q, X)\}$;
      for each $X \in \mathcal{R}$ do
        if $X$ is unclassified or noise then
          assign current clusterID to $X$
        if $X$ is unclassified then
          insert $X$ into $\Phi$;
      remove $Q$ from $\Phi$;

STEP 2.2. if not $\textsc{Core}^{cor}_{den}(O)$ $O$ is noise:

    mark $O$ as noise;

end.
```

Figure 8.5: Pseudo code of the 4C algorithm.

mensionality. The idea of our correlation connected clustering method is founded on DBSCAN, a density based clustering algorithm for Euclidean data spaces. The complexity of the original DBSCAN algorithm depends on the existence of an index structure for high dimensional data spaces. The worst case complexity is $O(n^2)$, but the existence of an efficient index reduces the complexity to $O(n \log n)$ [47]. DBSCAN is linear in the dimensionality of the data set for the Euclidean distance metric. If a quadratic form distance metric is applied instead of Euclidean (which enables user adaptability of the distance function), the time complexity of DBSCAN is $O(d^2 \cdot n \log n)$.

We begin our analysis with the assumption of no index structure.

Our algorithm has to associate each point of the data set with a similarity measure that is used for searching neighbors (cf. Definition 8.10). We assume that the corresponding similarity matrix must be computed once for each point, and it can be held in the cache until it is no more needed (it can be easily decided whether or not the similarity matrix can be safely discarded). The covariance matrix is filled with the result of a Euclidean range query which can be evaluated in $O(d \cdot n)$ time. Then the matrix is decomposed using PCA which requires $O(d^3)$ time. For all points together, we have $O(d \cdot n^2 + d^3 \cdot n)$.

Checking the correlation core point property according to Definition 8.12, and expanding a correlation connected cluster requires for each point the evaluation of a range query with a quadratic form distance measure which can be done in $O(d^2 \cdot n)$. For all points together (including the above cost for the determination of the similarity matrix), we obtain an worst-case time complexity of $O(d^2 \cdot n^2 + d^3 \cdot n)$.

Under the assumption that an efficient index structure for high dimensional data spaces (e.g. [30, 27]) is available, the complexity of all range queries is reduced from $O(n)$ to $O(\log n)$. Let us note that we can use Euclidean range queries as a filter step for the quadratic form range queries because no semi-axis of the corresponding ellipsoid exceeds ε. Therefore, the overall time complexity in this case is $O(d^2 \cdot n \log n + d^3 \cdot n)$.

8.2.3 Input Parameters

The algorithm 4C needs four input parameters which are discussed in the following:

The parameter $\varepsilon \in \mathbb{R}^+$ specifies the size of the local areas in which the correlations are examined and thus determines the number of objects which contribute to the covariance matrix and consequently to the correlation similarity measure of each object. It also participates in the determination of the density threshold, a cluster must exceed. Its choice usually depends on the volume of the data space (i.e. the maximum value of each attribute and the dimensionality of the feature space). The choice of ε has two aspects. First, it should not be too small because in that case, an insufficient number of objects contribute to the correlation similarity measure of each object and thus, this measure can be meaningless. On the other hand, ε should not be too large because then some noise objects might be correlation reachable from objects within a correlation connected cluster. Let us note, that our experiments indicated that the second aspect is not significant for 4C (in contrast to ORCLUS).

The parameter *MinPts* $\in \mathbb{N}$ specifies the number of neighbors an object must find in an ε-neighborhood and in a correlation ε-neighborhood to exceed the density threshold. It determines the minimum cluster size. The choice of *MinPts* should not be to small (*MinPts* ≥ 5 is a reasonable lower bound) but is rather insensitive in a broad range of values.

Both ε and *MinPts* should be chosen hand in hand.

The parameter $\lambda \in \mathbb{N}$ specifies the correlation dimension of the correlation connected clusters to be computed. As discussed above, the correlation dimension of a correlation connected cluster corresponds to its intrinsic dimension. In our experiments, it turned out that λ can be seen as an upper bound for the correlation dimension of the detected correlation connected clusters. However, the computed clusters tend to have a correlation dimension close to λ.

The parameter $\delta \in \mathbb{R}$ (where $0 \leq \delta \leq 1$) specifies the lower bound for the decision whether an eigenvalue is set to 1 or to $\kappa \gg 1$. The choice of δ influences the tightness of the detected correlations, i.e. how much local variance from the correlation is allowed. Our experiments also showed that $\delta \leq 0.1$ is usually a good choice.

8.3 Evaluation

In this section, we present a broad evaluation of 4C. We implemented 4C as well as the comparative methods DBSCAN and ORCLUS in JAVA. All experiments were run on a Linux workstation with a 2.0 GHz CPU and 2.0 GB RAM.

8.3.1 Efficiency

According to Section 8.2.2 the runtime of 4C scales superlinear with the number of input records. This is illustrated in Figure 8.6 showing the results of 4C applied to synthetic 2-dimensional data of variable size.

8.3.2 Effectiveness

We evaluated the effectiveness of 4C on several synthetic data sets as well as on real world data sets including gene expression data and metabolome data. In addition, we compared the quality of the results of our method to the quality of the results of DBSCAN and ORCLUS. In all our experiments, we set the parameter $\kappa = 50$ as suggested in Section 8.1.3.

Synthetic Data Sets

We first applied 4C on several synthetic data sets (with $2 \leq d \leq 30$) consisting of several dense, linear correlations. In all cases, 4C had no problems to identify the correlation-connected clusters. As an example, Figure 8.7 illustrates the parallel coordinate plot of the three clusters and the noise 4C found

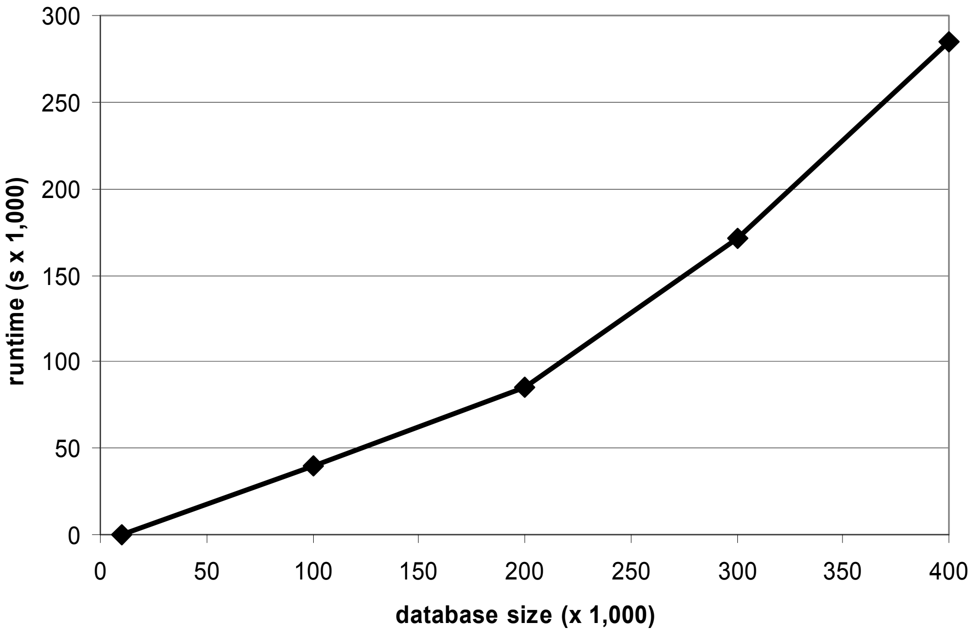


Figure 8.6: Scalability against database size.

on a sample 10-dimensional synthetic data set consisting of approximately 1,000 points.

Real World Data Sets

Gene Expression Data. We applied 4C to the gene expression data set of [134]. The data set is derived from time series experiments on the yeast mitotic cell cycle. The expression levels of approximately 3000 genes are measured at 17 different time slots. Thus, we face a 17-dimensional data space to search for correlations indicating co-regulated genes. 4C found 60 correlation connected clusters with few co-regulated genes (10-20). Such small cluster sizes are quite reasonable from a biological perspective. The parallel coordinate plots of four sample clusters are depicted in Figure 8.8. All four clusters exhibit simple linear correlations on a subset of their attributes. Let us note, that we also found other linear correlations which are rather complex to visualize.

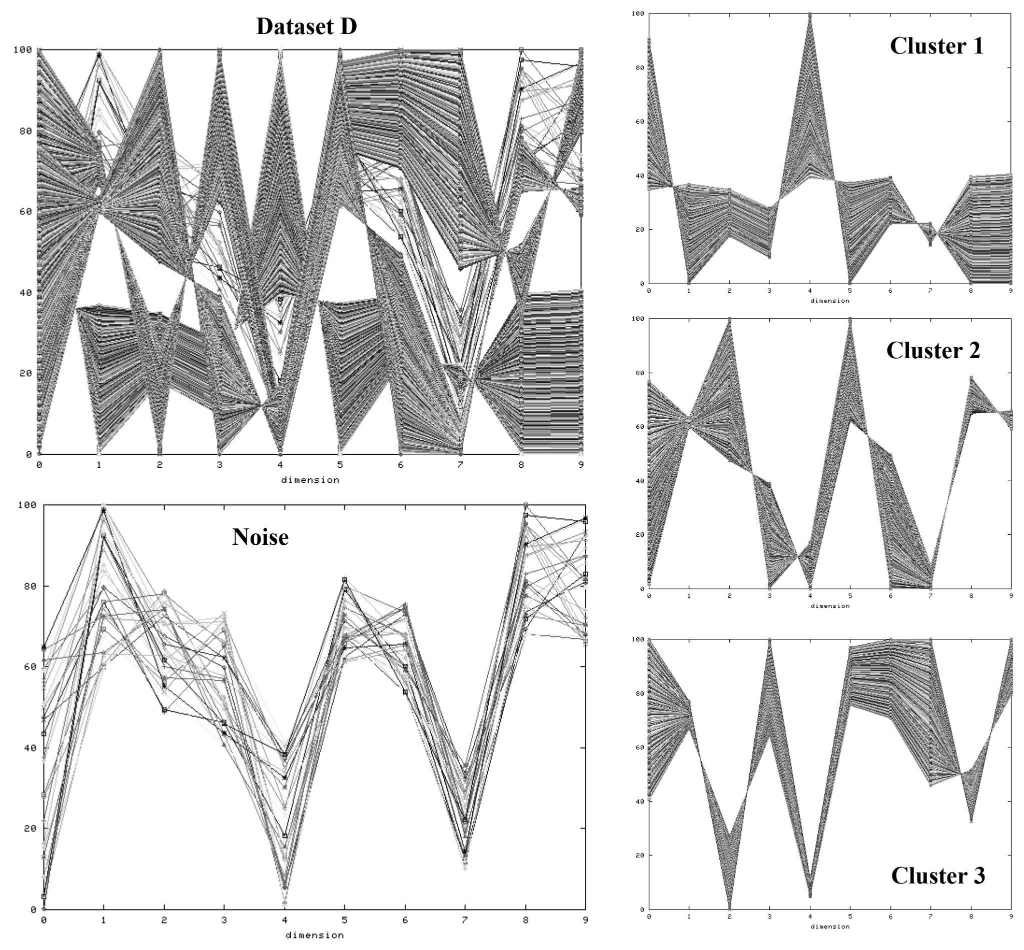


Figure 8.7: Clusters found by 4C on 10D synthetic data set. Parameters: $\varepsilon = 10.0$, $\mathit{MinPts} = 5$, $\lambda = 2$, $\delta = 0.1$.

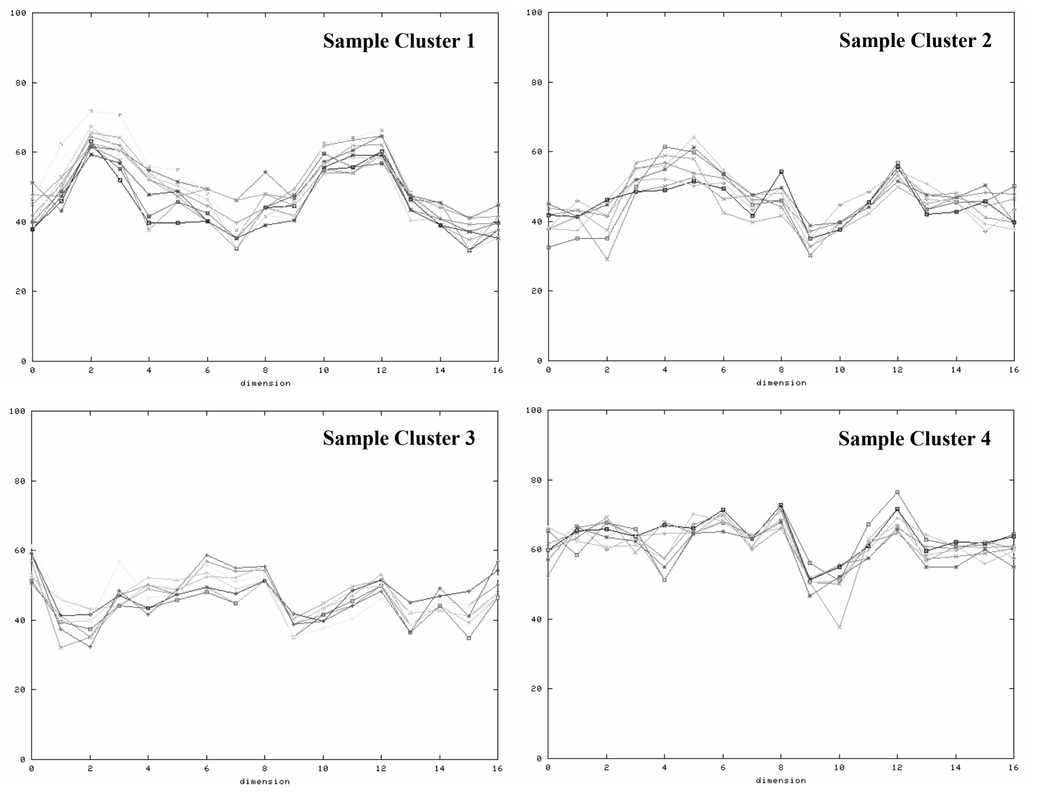


Figure 8.8: Sample clusters found by 4C on the gene expression data set. Parameters: $\varepsilon = 25.0$, $MinPts = 8$, $\lambda = 8$, $\delta = 0.01$.

Metabolome Data. We applied 4C on a metabolome data set [92]. The data set consists of the concentrations of 43 metabolites in 2,000 human newborns. The newborns were labeled according to some specific metabolic diseases. Thus, the data set consists of 2,000 data points with $d = 43$. 4C detected six correlation connected sets which are visualized in Figure 8.9. Cluster one and two (in the lower left corner marked with "control") consists of healthy newborns whereas the other clusters consists of newborns having one specific disease (e.g. "PKU" or "LCHAD"). The group of newborns suffering from "PKU" was split in three clusters. Several ill as well as healthy newborns were classified as noise.

Comparisons to Other Methods

We compared the effectiveness of 4C with related clustering methods, in particular the density-based clustering algorithm DBSCAN and the projected clustering algorithm ORCLUS. For that purpose, we applied these methods on several synthetic data sets including 2-dimensional data sets and higher dimensional data sets ($d = 10$).

Comparison with DBSCAN. The clusters found by DBSCAN and 4C applied on the 2-dimensional data sets are depicted in Figure 8.10. In both cases, DBSCAN finds clusters which do not exhibit correlations (and thus are not detected by 4C). In addition, DBSCAN cannot distinguish varying correlations which overlap (e.g. both correlations in data set B in Figure 8.10) and treat such clusters as one density-connected set, whereas 4C can differentiate such correlations. We gain similar observations when we applied DBSCAN and 4C on the higher dimensional data sets. Let us note, that these results are not astonishing since DBSCAN only searches for density connected sets but does not search for correlations and thus cannot be applied to the task of finding correlation connected sets.

Comparison with ORCLUS. A comparison of 4C with ORCLUS resulted in quite different observations. In fact, ORCLUS computes clusters of

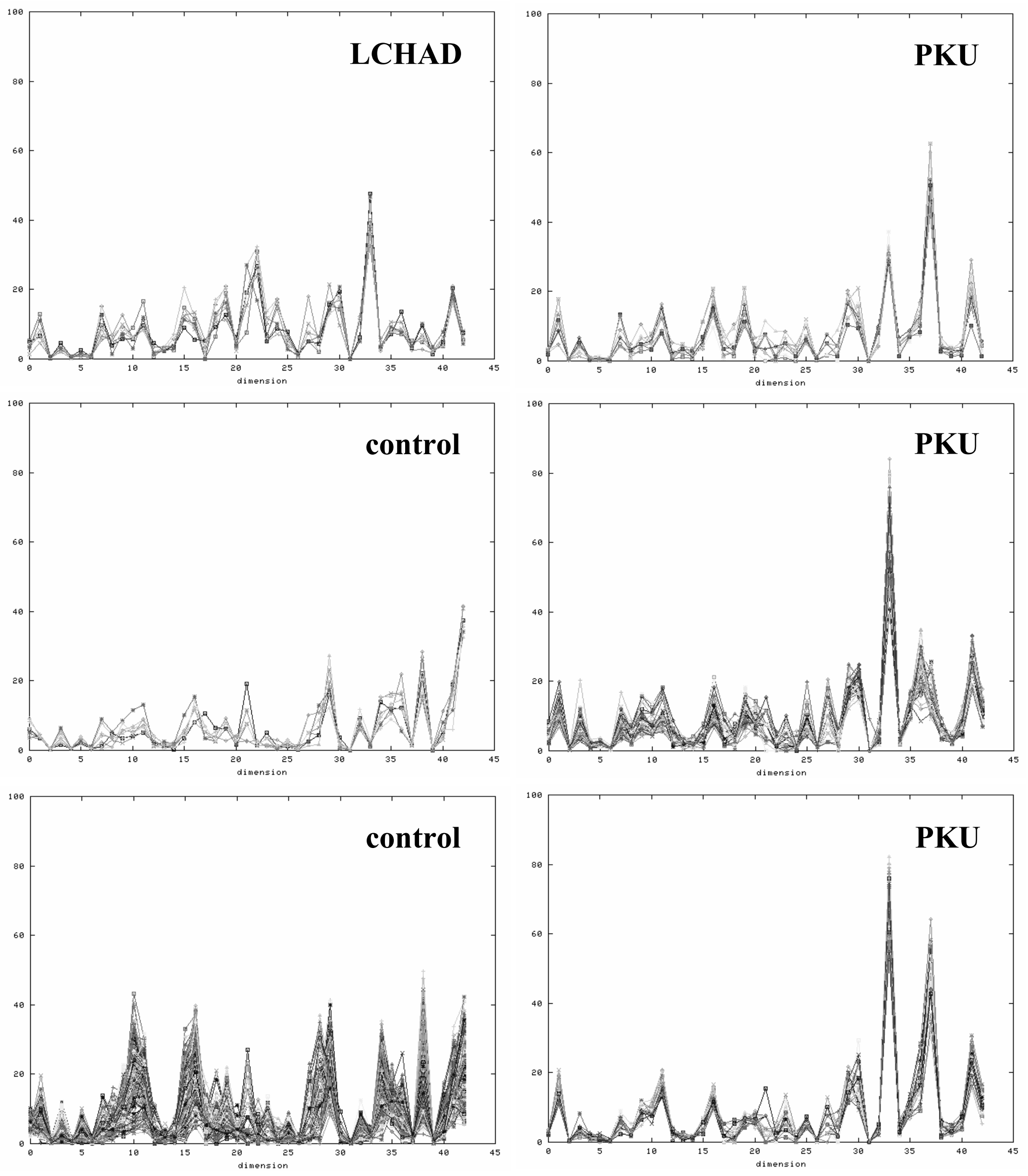


Figure 8.9: Clusters found by 4C on the metabolome data set. Parameters: $\varepsilon = 150.0$, $MinPts = 8$, $\lambda = 20$, $\delta = 0.1$.

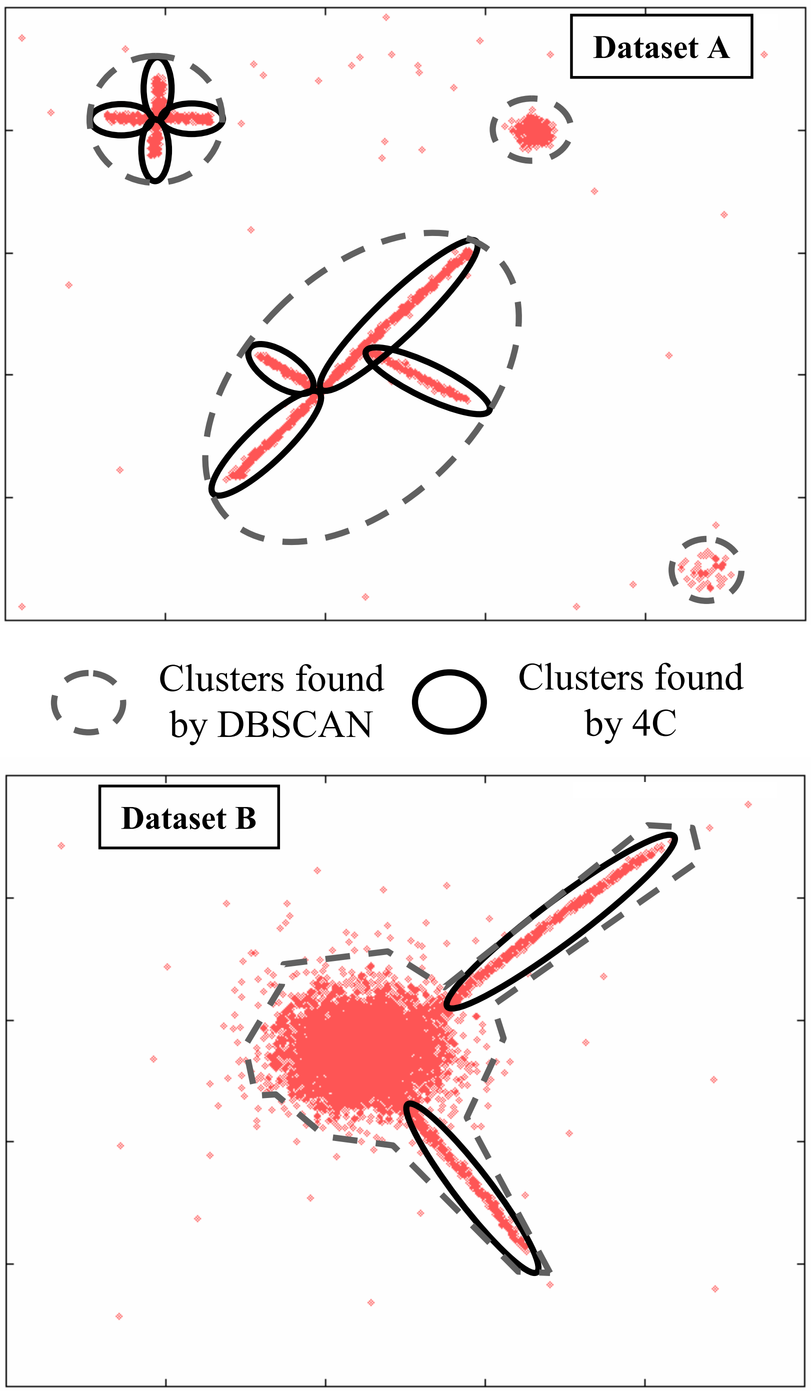


Figure 8.10: Comparison between 4C and DBSCAN.

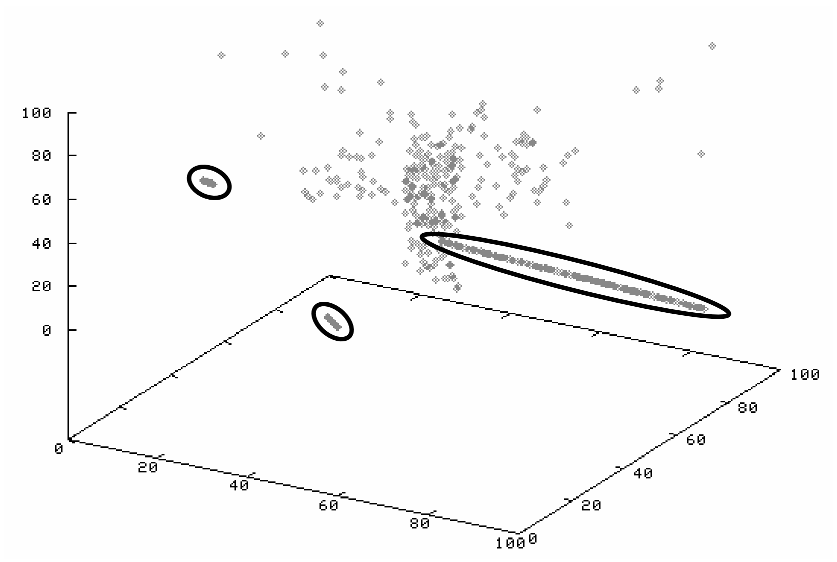


Figure 8.11: Three correlation connected clusters found by 4C on a 3-dimensional data set. Parameters: $\varepsilon = 2.5$, $MinPts = 8$. $\delta = 0.1$, $\lambda = 2$.

correlated objects. However, since it is a k-medoid based, it suffers from the following two drawbacks: First, the choice of k is a rather hard task for real-world data sets. Even for synthetic data sets, where we knew the number of clusters beforehand, ORCLUS often performs better with a slightly different value of k. Second, ORCLUS is rather sensitive to noise which often appears in real-world data sets. Since all objects have to be assigned to a cluster, the locality of the analyzed correlations is often too coarse (i.e. the subsets of the points taken into account for correlation analysis are too large). As a consequence, the correlation clusters are often blurred by noise objects and thus are hard to obtain from the resulting output. Figure 8.11 illustrates a sample 3-dimensional synthetic data set, the clusters found by 4C are marked by black lines. Figure 8.12 depicts the objects in each cluster found by ORCLUS ($k = 3$ yields the best result) separately. It can be seen, that the correlation clusters are — if detected — blurred by noise objects. When we applied ORCLUS on higher dimensional data sets ($d = 10$) the choice of k became even more complex and the problem of noise objects blurring the clusters (i.e. too coarse locality) simply cumulated in the fact that ORCLUS often could not detect correlation clusters in high dimensional data.

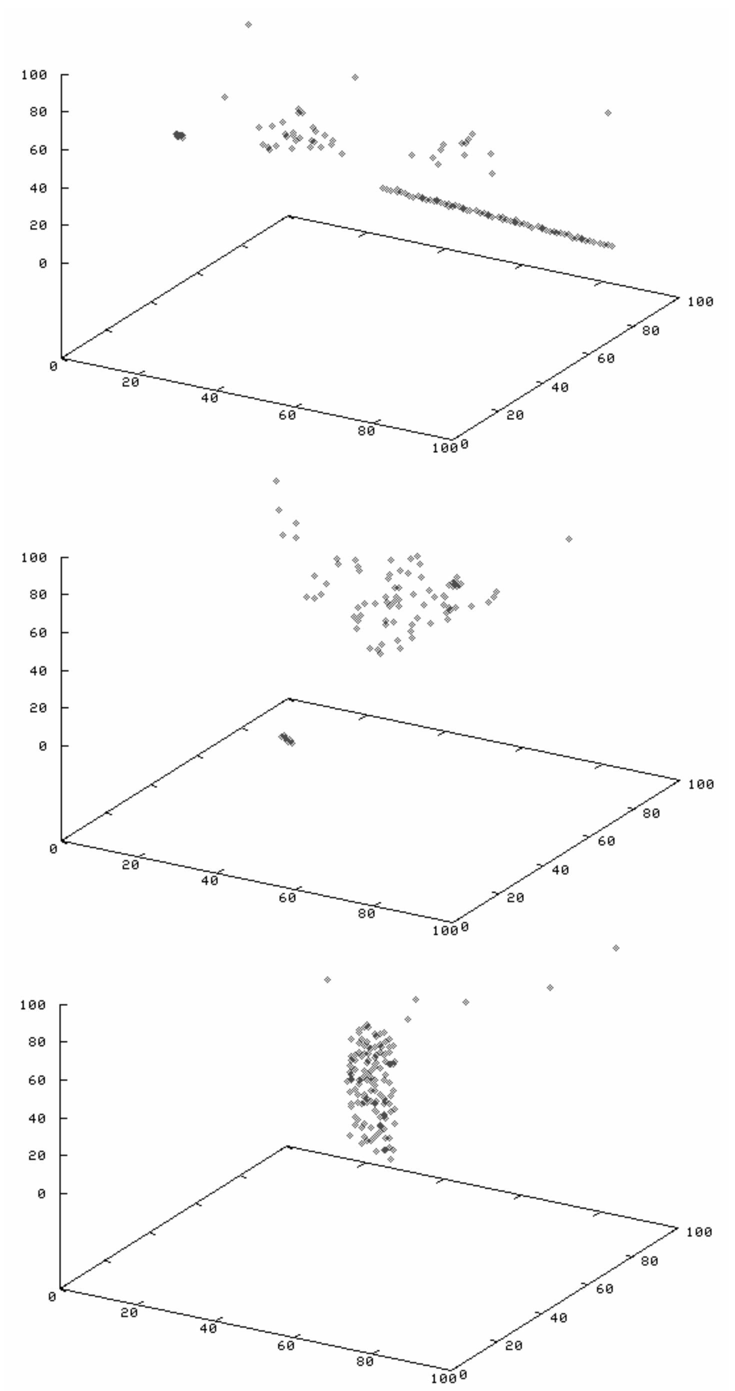


Figure 8.12: Clusters found by ORCLUS on the data set depicted in Figure 8.11. Parameters: $k = 3$, $l = 2$.

Chapter 9

Enhancing Efficiency and Effectiveness: COPAC

Correlation clusters appear as lines, planes, or, generally speaking, hyperplanes of arbitrary dimensionality $d_i < d$ in the data space, exhibiting a relatively high density of data points compared to the surrounding space. Correlation clustering algorithms group the data sets into subsets called correlation clusters such that the objects in the same correlation cluster are all associated to the same hyperplane of arbitrary dimensionality. For sake of brevity, if we have a correlation cluster associated to a λ-dimensional hyperplane, we will speak of a λ-dimensional correlation cluster. We will refer to the dimensionality of a hyperplane associated to a correlation cluster as correlation dimensionality. Of course, in applying correlation clustering one must be aware that, although linear correlation among features may indicate linear dependencies, the detected correlations can also be caused by features not comprised in the data set or they may even occur coincidentally.

Algorithms for correlation clustering integrate the concepts of clustering and correlation detection in a sophisticated way. The first approach that is exclusively designed to detect correlation clusters is ORCLUS [11] that integrates PCA into k-means clustering. The algorithm 4C [35] that integrates PCA into a density-based clustering algorithm shows superior effectiveness over ORCLUS (see Chapter 8).

However, existing correlation clustering methods have several severe problems. The most important problem is that the correlation dimensionality of the detected correlation clusters heavily depends on a user-defined input parameter. ORCLUS generates results such that the correlation dimensionality of the respective correlation clusters corresponds to a user-provided parameter l. 4C limits the correlation dimension of the detected correlation clusters to the user-defined parameter λ. In fact, 4C tends to uncover correlation clusters of a correlation dimensionality that is rather near to λ. As a consequence, both methods may produce incomplete results, i.e. both are not able to find all correlation clusters of different correlation dimensionality during a single run, especially if the correlation dimensionalities of different correlation clusters vary considerably. A second drawback related to the first problem is the poor usability of the existing methods because they require the user to specify parameters that are usually hard to determine, e.g. the number of clusters, or the "thickness" of the correlation hyperplane. Further limitations of existing work include a weak robustness against noisy data and a poor scalability for large databases.

The most straightforward possibility to overcome the first limitation is to apply one of the existing algorithms multiple times (in fact $O(d)$ times, where d is the dimensionality of the feature space). Obviously, this is not a reasonable solution due to the considerable high computational cost for one single run. In this chapter, we propose the novel correlation clustering algorithm COPAC (COrrelation PArtition Clustering) that simultaneously searches for correlation clusters of arbitrary dimensionality. Let us point out that COPAC does not require the user to specify the number of clusters or any parameter regarding the correlation dimensionality beforehand. In order to further enhance the usability of COPAC, we will discuss the effect of the input parameters of COPAC. In addition, our experimental evaluation shows that COPAC is superior to ORCLUS and 4C in terms of runtime and produces significantly more accurate and complete results.

This chapter is organized as follows: We present a formalization of correlation clusters in Section 9.1. Section 9.2 describes the algorithm COPAC in detail. A thorough experimental evaluation of COPAC (Section 9.3) demonstrates that COPAC is superior to ORCLUS and 4C in terms of runtime and produces significantly more accurate and complete results.

The material presented in this chapter has been partially published in [6].

9.1 Formalization of Correlation Clusters

In this section, we prepare the introduction of our approach by formalizing the notion of correlation clusters. In the following we assume $\mathcal{D}$ to be a database of n feature vectors in a d-dimensional feature space, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$. A correlation cluster is a set of feature vectors that are close to a common, arbitrarily oriented affine subspace of a given dimensionality d_i ($1 \leq d_i < d$). In the data space the correlation cluster appears as a hyperplane of dimensionality d_i.

In general, one way to formalize the concept of correlation clusters is to use PCA. Formally, let $\mathcal{C}$ be a correlation cluster, i.e. $\mathcal{C} \subseteq \mathcal{D}$, and $\bar{X}$ denote the centroid of all points in $\mathcal{C}$. The $d \times d$ *covariance matrix* $\boldsymbol{\Sigma}_{\mathcal{C}}$ of $\mathcal{C}$ is defined as:

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

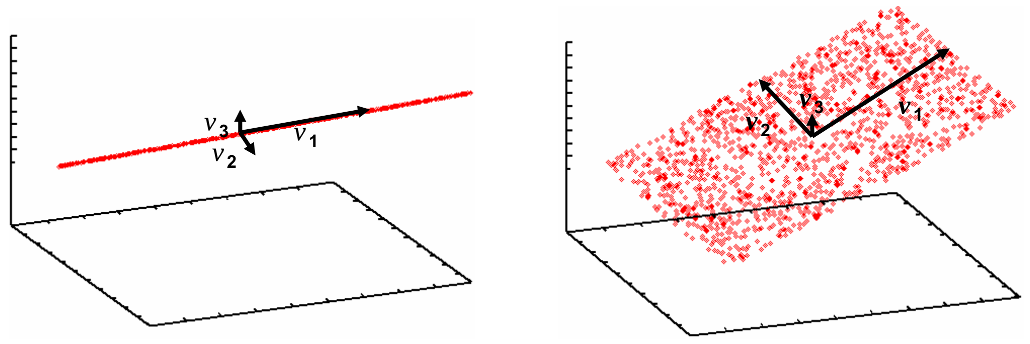
Since the covariance matrix $\boldsymbol{\Sigma}_{\mathcal{C}}$ of $\mathcal{C}$ is a positive semi-definite square matrix, it can be decomposed into the *eigenvalue matrix* $\boldsymbol{E}_{\mathcal{C}}$ of $\boldsymbol{\Sigma}_{\mathcal{C}}$ and the *eigenvector matrix* $\boldsymbol{V}_{\mathcal{C}}$ of $\boldsymbol{\Sigma}_{\mathcal{C}}$ such that $\boldsymbol{\Sigma}_{\mathcal{C}} = \boldsymbol{V}_{\mathcal{C}} \cdot \boldsymbol{E}_{\mathcal{C}} \cdot \boldsymbol{V}_{\mathcal{C}}^{\mathsf{T}}$. The eigenvalue matrix $\boldsymbol{E}_{\mathcal{C}}$ is a diagonal matrix storing the d non-negative eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{C}}$ in decreasing order. The eigenvector matrix $\boldsymbol{V}_{\mathcal{C}}$ is an orthonormal matrix with the corresponding d eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{C}}$.

Now we define the correlation dimensionality of $\mathcal{C}$ as the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes in $\boldsymbol{V}_{\mathcal{C}}$. Let us 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 by a common line, the correlation dimensionality of these points will be 1. That means we have to determine the principal components (eigenvectors) of $\boldsymbol{\Sigma}_{\mathcal{C}}$. 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 $\boldsymbol{\Sigma}_{\mathcal{C}}$ 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 $\alpha \in]0,1[$ of the total variance of $\mathcal{C}$. These ideas are illustrated in Figure 9.1. Figure 9.1(a) shows a correlation cluster of correlation dimensionality 1 corresponding to a *correlation line.* Only one eigenvector (v_1) explains the total variance of $\mathcal{C}$. Figure 9.1(b) shows a correlation cluster of correlation dimensionality 2 that corresponds to a *correlation plane.* Here, two eigenvectors explain the total variance of $\mathcal{C}$. Let us note that in the displayed examples, the correlations are perfect, i.e. there is no deviation from the correlation hyperplane but all points within the set perfectly fit to the correlation hyperplane. As a consequence, the eigenvalues of the eigenvectors that are orthogonal to the correlation hyperplane (e.g. v_2 and v_3 in Figure 9.1(a) or v_3 in Figure 9.1(b)) will be zero. However, in real-world data sets, this is a quite unrealistic scenario, i.e. the eigenvalues of some eigenvectors that are orthogonal to the correlation hyperplane may be considerably small but not zero. The value of α accounts for that fuzziness. Let us define the correlation dimensionality more formally:

Definition 9.1 (correlation dimensionality)
Let $\alpha \in]0,1[$. Then the correlation dimensionality *$\lambda_{\mathcal{C}}$ of a set of points $\mathcal{C}$ is the smallest number r of eigenvalues e_i in the $d \times d$ eigenvalue matrix $\boldsymbol{E}_{\mathcal{C}}$ 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 call the $\lambda_{\mathcal{C}}$-dimensional subspace which



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

Figure 9.1: Correlation dimensionality.

is spanned by the major axes of $\mathcal{C}$ the *correlation hyperplane* of $\mathcal{C}$. Since we follow the convention that the eigenvectors are ordered decreasingly in the eigenvector matrix, the major axes correspond to the $\lambda_{\mathcal{C}}$ first eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{C}}$.

Thus, the correlation dimensionality $\lambda_{\mathcal{C}}$ is the dimensionality of the subspace containing all points of the set $\mathcal{C}$ allowing a small deviation corresponding to the remaining portion of variance of $1 - \alpha$. The remaining, neglected variance scatters along the eigenvectors $v_{\lambda_{\mathcal{C}}+1}, \ldots, v_d$.

9.2 COPAC

As discussed above, in correlation clustering algorithms like ORCLUS and 4C the user needs to estimate an appropriate correlation dimensionality in advance. If this estimation is wrong, the quality of the derived clustering deteriorates considerably. An alternative would be, of course, to run the respective algorithms several times, each time using another guess for the correlation dimensionality. We propose an approach that does not require a parameter specifying the correlation dimensionality and searches the data for clusters of all possible correlation dimensionalities simultaneously. It is possible, as we will show, to perform this search in superior or at least competitive average efficiency even if compared to *single* runs of ORCLUS, that will uncover only a specified number of clusters of a predefined correlation dimensionality. Thus, COPAC can retrieve superior information in equal or less runtime than ORCLUS or 4C.

The general idea of COPAC is as follows: We adapt the definition of correlation cluster dimensionality (cf. Definition 9.1) as a property of single database objects resulting in the notion of *local* correlation dimensionality. We partition the database objects according to their local correlation dimensionality in a first step. The local correlation dimensionality of a point P represents the cluster correlation dimensionality of the set of points in the neighborhood of P in the database, i.e. the correlation dimensionality of the correlation cluster—if existing—P should belong to. 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. Therefore, in a second step, we apply a novel correlation clustering algorithm to each of the partitions in order to compute correlation clusters of different correlation dimensionalities. In the following, we describe both steps in more detail.

9.2.1 Local Correlation Partitioning

In the first step of COPAC we partition the objects of the database according to their local correlation dimensionality. The local correlation dimensionality is defined analogously to Definition 9.1.

Definition 9.2 (local correlation dimensionality)
Let $\alpha \in]0,1[$, $P \in \mathcal{D}$, and $\mathcal{N}_P$ denote the set of points in the local neighborhood of P. $\boldsymbol{E}_{\mathcal{N}_P}$ is the eigenvalue matrix of $\boldsymbol{\Sigma}_{\mathcal{N}_P}$ which is the covariance matrix of $\mathcal{N}_P$. Then the local correlation dimensionality λ_P *of the point P is the smallest number of eigenvalues e_i in the eigenvalue matrix $\boldsymbol{E}_{\mathcal{N}_P}$ explaining a portion of at least α of the total variance, i.e.*

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

Again, values for α typically range from 0.8 to 0.9.

An important aspect of Definition 9.2 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 [35] (see Chapter 8), 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 $\boldsymbol{\Sigma}_{\mathcal{N}_P}$ 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 [35] (see Chapter 8) 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, we use 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 $\boldsymbol{\Sigma}_{\mathcal{N}_P}$ and a stable singular value decomposition of $\boldsymbol{\Sigma}_{\mathcal{N}_P}$ 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 using k-nearest neighbors, while the ε-neighborhood would provide a considerably larger amount of points resulting in a local correlation dimensionality that does not reflect the actual local correlation dimensionality as exactly as the k-nearest neighbors do.

Still, however, the value for k is chosen globally for all points and may be suitable for some local neighborhoods (and some clusters), for others not. This problem is addressed in another approach which we will discuss in Chapter 12 below.

Based on Definition 9.2, the first step of COPAC partitions the database objects according to their local correlation dimensionality. The local correlation dimensionality of a point is based on the covariance matrix for the k-nearest neighbors of that point in $\mathcal{D}$. Then all points P sharing a common local correlation dimensionality λ_P are assigned to a partition $\mathcal{D}_{\lambda_P}$ of the database $\mathcal{D}$. This results in a set of d disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_d$ of $\mathcal{D}$, where $\mathcal{D}_i$ contains all points exhibiting a correlation dimensionality of i. Of course, some partitions 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, because if $\lambda_P = d$, then there is no linear dependency of features found among the neighbors of P. Note that the number of attributes actually involved in linear dependencies within cluster $\mathcal{C}$ is not $\lambda_{\mathcal{C}}$, but $d - \lambda_{\mathcal{C}}$.

Let us note that by means of this first step of COPAC one does not only yield an appropriate correlation dimensionality for each point in advance, but presumably also a considerable reduction of the number of data points n, that are to be processed by each single run of a correlation clustering algorithm, on average $\frac{n}{d}$. In fact, COPAC processes each data object only once during the second step when determining the correlation clusters.

9.2.2 Determination of Correlation Clusters

Once we have partitioned the database objects into partitions $\mathcal{D}_1, \ldots, \mathcal{D}_d$ according to their local correlation dimensionality in $\mathcal{D}$, we can extract the correlation clusters from each of the partitions. Since each point $P \in \mathcal{D}$ can only be part of a correlation cluster of dimensionality λ_P, we can run the correlation cluster extraction on each partition $\mathcal{D}_1, \ldots, \mathcal{D}_{d-1}$ separately. As discussed above, the points in $\mathcal{D}_d$ are noise because there is no linear dependency among a set of features in the local neighborhood of these points.

To detect the correlation clusters in a given partition $\mathcal{D}_i$, we can use any correlation clustering algorithm proposed so far. Since 4C has shown superior effectiveness over ORCLUS (see Chapter 8), we base our method to compute the correlation clusters on the concepts of 4C. The most important aspect of applying density based clustering is that each partition $\mathcal{D}_i$ may contain noise points that have a local correlation dimensionality of i but do not belong to any correlation cluster. In fact, 4C is reported to be much more robust against noise than ORCLUS.

However, our method is considerably different from 4C in two aspects: First, in contrast to 4C (and also to ORCLUS), we can restrict our method to find only correlation clusters of a given correlation dimensionality, because the points in $\mathcal{D}_i$ can only be part of a correlation cluster with correlation dimensionality i. Second, 4C limits itself to "connected" correlation clusters, i.e. points that share a common hyperplane but are located significantly far apart are not assigned to the same cluster. Our method overcomes this limitation.

The general idea of our correlation clustering method is to integrate PCA as a correlation primitive into the density-based clustering algorithm DBSCAN [47]. For that purpose we define a distance measure that assesses the distance between two given points evaluating how well they share a common hyperplane. The Euclidean distance between the respective points is assessed only to measure the deviation orthogonal to the common hyperplane. To yield such a distance measure we first define a distance between two points with respect to one of the points. This distance is based on a distance matrix for each point P that is derived by an adaptation of the eigenvalues of the covariance matrix of the local neighborhood of P:

Definition 9.3 (correlation distance matrix)
Let $P \in \mathcal{D}$, λ_P the local correlation dimensionality of P, and $\boldsymbol{V}_P$, $\boldsymbol{E}_P$ the corresponding eigenvectors and eigenvalues of the point P based on the local neighborhood of P, i.e. $\mathcal{N}_P$. An adapted eigenvalue matrix $\hat{\boldsymbol{E}}_P$ with diagonal entries $\hat{e}_i \in \{0, 1\}$, $(i = 1, \ldots, d)$ is derived according to the following rule:

$$\hat{e}_i = \begin{cases} 0 & \textit{if} \quad i \leq \lambda_P \\ 1 & \textit{if} \quad i > \lambda_P \end{cases}$$

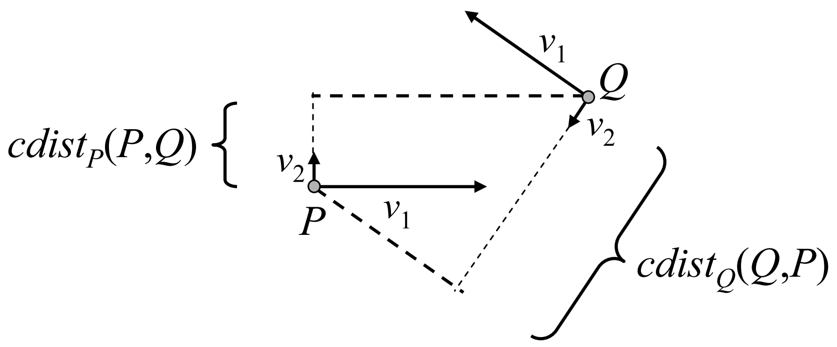


Figure 9.2: Correlation distance measure of a point.

The matrix $\hat{\boldsymbol{M}}_P = \boldsymbol{V}_P \cdot \hat{\boldsymbol{E}}_P \cdot \boldsymbol{V}_P^{\mathsf{T}}$ *is called the* correlation distance matrix *of* P.

Using the *correlation distance matrix of* P one can easily derive a distance measure that assesses the distance between P and another point Q w.r.t. P:

Definition 9.4 (correlation distance measure)
Let $P, Q \in \mathcal{D}$. *The* correlation distance measure *between* P *and* Q *w.r.t. point* P *is given by:*

$$cdist_P(P, Q) = \sqrt{(P - Q)^{\mathsf{T}} \cdot \hat{\boldsymbol{M}}_P \cdot (P - Q)}.$$

Basically, the *correlation distance measure* w.r.t. a point P is a weighted distance where the weights are based on the local neighborhood of P. The weights are constructed to take into account only distances along the eigenvectors that correspond to small eigenvalues, while distances along the λ_P first eigenvectors of $\mathcal{N}_P$ are neglected. Thus, assessing the distance between P and Q using the *correlation distance measure* of P will in general not yield the same result as using the *correlation distance measure* w.r.t. Q. The concept of the correlation distance measure w.r.t. two points P and Q, with $\lambda_P = \lambda_Q = 1$, is visualized in Figure 9.2. As it can be seen, $cdist_P(P, Q) \neq cdist_Q(Q, P)$. Therefore, given the *correlation distance measures* for both points, P and Q, we define a distance function (i.e. a distance measure that fulfills symmetry and reflexivity) as follows:

Definition 9.5 (correlation distance)
Let $P, Q \in \mathcal{D}$. The correlation distance *between P and Q is given by:*

$$cdist(P,Q) = \max\{cdist_P(P,Q), cdist_Q(Q,P)\}$$

For example, the correlation distance for the points P and Q in Figure 9.2 is equal to the correlation distance measure between both points w.r.t. Q, i.e. $cdist(P,Q) = cdist_Q(Q,P)$.

Having defined a suitable distance function for correlation clustering, we can now integrate these concepts into a clustering algorithm. We propose to integrate the correlation distance into the density-based clustering algorithm GDBSCAN [120] which is a generalization of the well-known DBSCAN clustering algorithm [47]. The choice of GDBSCAN is because of its efficiency and its effectiveness. 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 *MinPts* 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 *MinPts* 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 $MinPts \in \mathbb{N}^+$ are the input parameters specifying the density threshold points within a cluster must exceed.

The GDBSCAN framework as proposed in [120] 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 O, denoted by $\mathcal{N}_{NPred}(O)$, given by $\mathcal{N}_{NPred}(O) = \{P \mid NPred(O,P)\}$, where $NPred(O,P)$ is a predicate on O and P that has to be reflexive and symmetric. In addition, to decide whether or not object O is a core point, a generalized minimum weight of $\mathcal{N}_{NPred}(O)$ must be defined, denoted by $MinWeight(\mathcal{N}_{NPred}(O))$.

Thus, in order to integrate our correlation distance measure into the GDBSCAN algorithm, we need to specify (i) a symmetric and reflexive predicate $NPred(P, Q)$ on two points $P, Q \in \mathcal{D}$ and (ii) a minimum weight *MinWeight*.

The key issue is the predicate $NPred(P, Q)$. Intuitively, we define this predicate analogously to the ε-neighborhood, using the correlation distance from Definition 9.5.

Definition 9.6 (neighborhood predicate)
Let $P, Q \in \mathcal{D}$ and $\varepsilon \in \mathbb{R}^+$. The neighborhood predicate *of P and Q is given by: $NPred(P, Q) \Leftrightarrow cdist(P, Q) \leq \varepsilon$.*

The neighborhood predicate $NPred(P, Q)$ is reflexive and symmetric, since it is based on the reflexive and symmetric correlation distance as defined in Def. 9.5. We show this formally in the following lemma.

Lemma 9.1
The neighborhood predicate $NPred(P, Q)$ as defined in Definition 9.6 is reflexive and symmetric.

Proof.
Let $P, Q \in \mathcal{D}$ and $\varepsilon \in \mathbb{R}^+$.

(i) reflexivity: clear since

$$\begin{aligned} NPred(P, P) &\Leftrightarrow cdist(P, P) \leq \varepsilon \\ &\Leftrightarrow cdist_P(P, P) \leq \varepsilon \\ &\Leftrightarrow 0 \leq \varepsilon. \end{aligned}$$

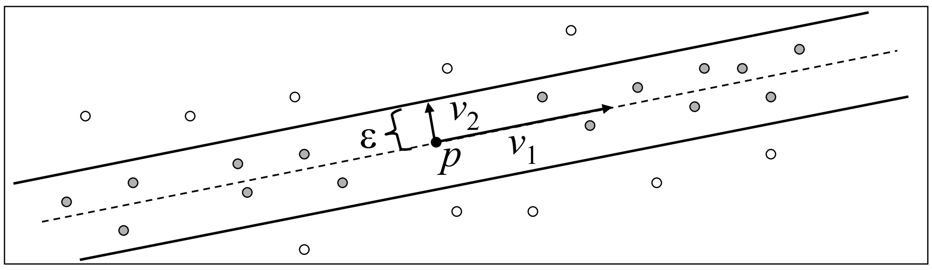


Figure 9.3: Visualization of $\mathcal{N}_{NPred}(P)$.

(ii) symmetry:

$$\begin{aligned}
NPred(P,Q) &\Leftrightarrow cdist(P,Q) \leq \varepsilon \\
&\Leftrightarrow \max\{cdist_P(P,Q), cdist_Q(Q,P)\} \leq \varepsilon \\
&\Leftrightarrow \max\{cdist_Q(Q,P), cdist_P(P,Q)\} \leq \varepsilon \\
&\Leftrightarrow cdist(Q,P) \leq \varepsilon \\
&\Leftrightarrow NPred(Q,P).
\end{aligned}$$

□

The neighborhood predicate is visualized in Figure 9.3. In particular, the figure shows the neighborhood of P, i.e. $\mathcal{N}_{NPred}(P)$. All objects Q within this neighborhood have a distance less or equal to ε to the correlation hyperplane specified by the correlation distance matrix of P. This hyperplane is indicated by the dashed line. Furthermore, if an object Q is a member of the neighborhood of P, then also P must have a distance less or equal to ε to the correlation hyperplane of Q. In other words, $Q \in \mathcal{N}_{NPred}(P)$ if and only if the correlation distance measure between P and Q w.r.t. P and w.r.t. Q both do not exceed ε.

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, we require that a point P finds at least *MinPts* points in its ε-neighborhood using the correlation distance as distance function.

```
algorithm COPAC
  // STEP 1: Partition data objects according to
  //         local correlation dimensionality
  //         (cf. Section 9.2.1)
    initialize $\mathcal{D}_1, \ldots, \mathcal{D}_d$ with $\mathcal{D}_i = \emptyset$;
    for each $P \in \mathcal{D}$ do
        compute $\lambda_P$ according to Definition 9.2;
        $\mathcal{D}_{\lambda_P} = \mathcal{D}_{\lambda_P} \cup \{P\}$;
    endfor
  // STEP 2: Extract clusters from each partition
  //         (cf. Section 9.2.2)
    for each $\mathcal{D}_i \in \mathcal{D}_1, \ldots, \mathcal{D}_{d-1}$ do
        GDBSCAN($\mathcal{D}_i$, $NPred(\varepsilon)$, $MinWeight(\mu)$);
    endfor
```

Figure 9.4: The COPAC algorithm.

Definition 9.7 (minimum weight)
Let $\mathcal{N}_{NPred}(P)$ be the neighborhood of P based on the neighborhood predicate as defined in Definition 9.6 and $\mu \in \mathbb{N}^+$. The minimum weight of $\mathcal{N}_{NPred}(P)$ such that $P \in \mathcal{D}$ 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, we can use the GDBSCAN framework to compute the correlation clusters in each partition. The overall procedure of COPAC is visualized in Figure 9.4: Step 1 partitions the database points according to their local correlation dimensionality.[1] Step 2 applies GDBSCAN with *NPred* and *MinWeight* as defined in Definitions 9.6 and 9.7, respectively.

9.2.3 Complexity Analysis

The preprocessing step of COPAC works for each point as follows: First a k-nearest neighbor query is performed, which has a complexity of $O(n)$ since

[1]Note that it is a recommendable procedure to normalize the data in order to derive an appropriate correlation dimensionality. We usually followed this procedure.

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 using PCA which requires $O(d^3)$ time. Thus, for all points together we have a time complexity of $O(n^2 + k \cdot d^2 \cdot n)$ in the first step of COPAC, since k must exceed d, as discussed above.

Applying GDBSCAN to the data set in the second step of COPAC results in a time complexity of $O(d^2 \cdot n^2)$. This is due to the fact, that the original GDBSCAN has a worst case complexity of $O(n^2)$ on top of the sequential scan. If a quadratic distance function is used, like in our algorithm, the time complexity of GDBSCAN increases to $O(d^2 \cdot n^2)$.

Thus, the overall worst-case time complexity of COPAC on top of the sequential scan of the data set is $O(k \cdot d^2 \cdot n + d^2 \cdot n^2)$. However, usually the data points are distributed over several partitions. In the best-case, the data points are uniformly distributed over all possible correlation dimensionalities, and all partitions will contain $\frac{n}{d}$ points. Thus, the best-case reduces the required runtime of the second step of COPAC to $O(n^2)$ and the overall time-complexity to $O(k \cdot d^2 \cdot n + n^2)$.

9.2.4 Parameter Estimation

COPAC has three input parameters the choice of which we discuss in the following.

Parameter k. The parameter $k \in \mathbb{N}^+$ specifies the number of points considered to compute the neighborhood $\mathcal{N}_P$ of a point $P \in \mathcal{D}$. From this neighborhood, the $d \times d$ covariance matrix $\boldsymbol{\Sigma}_P$ and, thus, the correlation dimensionality λ_P of P is computed. As discussed above, k should not be too small in order to produce a stable covariance matrix. For example, in order to model a λ-dimensional correlation, we need at least λ points that span the corresponding λ-dimensional hyperplane. On the other hand, it should not be too high in order to reflect only the local correlation. Oth-

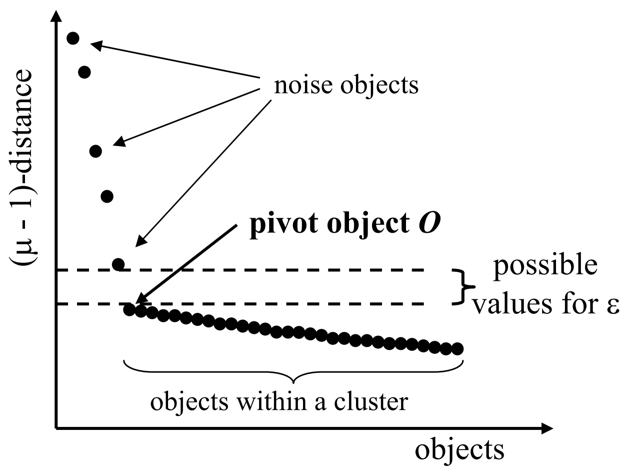


Figure 9.5: A sample $(\mu - 1)$-distance diagram.

erwise, noise points could also destabilize the correlation matrix and, thus, the computation of the local correlation dimensionality. It turned out that setting $k = 3 \cdot d$ was robust in all our tests throughout all our experiments. In general, setting $3 \cdot d \leq k$ seems to be a reasonable suggestion.

Parameter μ. The parameter $\mu \in \mathbb{N}^+$ specifies the minimal weight predicate on the neighborhood predicate in the GDBSCAN framework. In fact, μ specifies the minimum number of points in a cluster and, therefore, is quite intuitive. In general, the choice of μ depends on the application. Obviously, $\mu \leq k$ should hold.

Parameter ε. The parameter $\varepsilon \in \mathbb{R}^+$ is used to specify the neighborhood predicate in the GDBSCAN framework. In general, ε can be chosen as proposed first for the DBSCAN specialization [47] and also in [120]. The procedure depends on the choice of μ and works as follows. Given the parameter μ, we compute the $(\mu-1)$-nearest neighbor distances of all points in $\mathcal{D}$. These distances are plotted: the points are sorted according to decreasing $(\mu - 1)$-nearest neighbor distances along the x-axis. The corresponding $(\mu - 1)$-nearest neighbor distances are plotted along the y-axis. Figure 9.5 depicts such a sample $(\mu - 1)$-distance diagram. The main idea now is that the $(\mu-1)$-nearest neighbor distances of the noise points are scattered significantly, whereas the $(\mu-1)$-nearest neighbor distances of the points inside a correlation cluster should be rather similar. Thus, a plateau in the $(\mu-1)$-distance diagram indicates that the participating points form a correlation cluster. If we determine the first object O of the first plateau in the $(\mu-1)$-distance diagram, called the pivot object, we can extract O's $(\mu-1)$-distance and set ε to this value (cf. Figure 9.5). For performance reasons, we can compute the $(\mu-1)$-distance diagram only for a sample, e.g. only for 10% of the points in $\mathcal{D}$.

Let us note that we have in fact a fourth parameter α to compute the correlation dimensionality λ_P of a point $P \in \mathcal{D}$. This parameter specifies the portion of the total variance of the points which is explained by the λ_P greatest eigenvalues of the covariance matrix of P's neighborhood. As discussed above, this parameter is very robust in the range between $0.8 \leq \alpha \leq 0.9$. Thus, we choose $\alpha = 0.85$ throughout all our experiments.

9.3 Evaluation

In all our experiments, we choose the parameters for COPAC as suggested in Section 9.2.4. All competitors have been implemented within the framework ELKI [8].

9.3.1 Efficiency

For evaluation of efficiency, we used synthetic data sets where the number of points or the dimensionality has been varied. We created several randomized data sets based on a predefined distribution of random points over a given number of clusters. For the impact of the dimensionality of the data space on the runtime, we created 10 data sets with a dimensionality of $d = 10, 20, 30, \ldots, 100$. For each data set, 10,000 points were distributed over $d-1$ clusters of correlation dimensionality $\lambda = 1, \ldots, d-1$. Similarly we created 10 data sets of dimensionality $d = 10$ with an increasing number

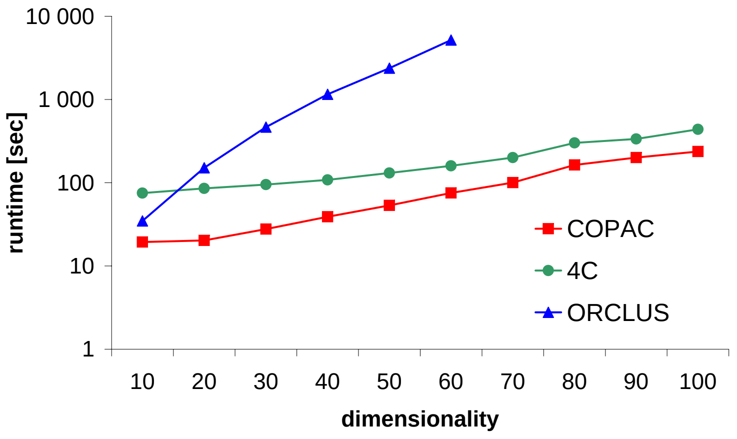


Figure 9.6: Runtime vs. data dimensionality.

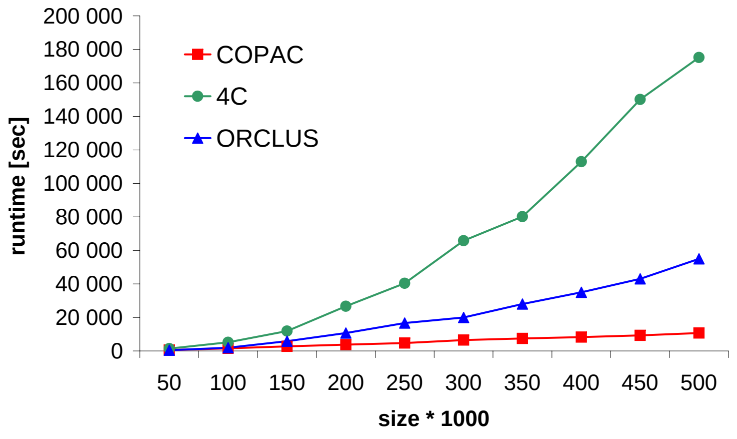


Figure 9.7: Runtime vs. data set size.

of points ranging from 50,000 to 500,000, distributed over several correlation clusters of correlation dimensionality $\lambda = 1, \ldots, 9$ and noise. Here all clusters are equally sized containing 2500 points.

Generally, the runtime complexity of 4C can be regarded as an upper-bound for the runtime-complexity of COPAC. The worst-case for COPAC occurs when all points share a common correlation dimensionality and, thus, the partitioning provides only one partition containing the complete data set. The performance of COPAC for such data would be comparable to the performance of 4C in terms of efficiency. But even if the case occurs that all points have an identical correlation dimensionality λ, the gain of COPAC is the additional information that no clusters with a correlation dimensionality different from λ are present in the data. This information is not given after the corresponding run of 4C. However, usually the data will be distributed over several partitions, thus the runtime decreases considerably, as experimentally demonstrated.

In our first experiment we compared the runtime of COPAC, 4C, and ORCLUS w.r.t. the dimensionality d of the data set. As parameters for COPAC we used $\varepsilon = 0.02$, $\mu = k = 100$ and $\alpha = 0.85$. Since the number of points within a cluster reaches its minimum for $d = 100$ we set the parameter μ and k to this minimum value. As a fair setting we gave as parameter k to ORCLUS the exact number of clusters in the data set and parameter l was set to the maximal occurring correlation dimensionality, i.e. $k = d$ and $l = d - 1$. The parameters for 4C were set to $\varepsilon = 0.1$, $\mu = 100$, $\lambda = d - 1$ and $\delta = 0.01$. As it can be seen in Figure 9.6, COPAC gains a significant speed-up over ORCLUS and 4C (note the logarithmic scale).

In our second experiment we evaluated the impact of the size of the data set on the runtime of COPAC, 4C, and ORCLUS. The parameters for COPAC were set to $\varepsilon = 0.02$, $\mu = k = 3 \cdot d = 30$ and $\alpha = 0.85$. As before, the parameter k of ORCLUS was set to the exact number of clusters in the data set and parameter l was set to the maximal occurring correlation dimensionality. As parameters for 4C we used analogously $\varepsilon = 0.1$, $\mu = 3 \cdot d = 30$, $\lambda = d - 1 = 9$ and $\delta = 0.01$. Figure 9.7 illustrates the runtime of COPAC, 4C, and ORCLUS w.r.t. the data set size. Again, COPAC clearly outperforms ORCLUS and 4C in terms of efficiency.

9.3.2 Robustness, Completeness, and Usability

To demonstrate the robustness, completeness, and usability of COPAC in comparison to ORCLUS and 4C, we synthesized a data set $\mathcal{D} \in \mathbb{R}^3$ with two clusters of correlation dimensionality 2, three clusters of correlation dimensionality 1, and some points of noise. The data set is depicted in Figure

9.8(a). The clusters partially intersect making the separation of the clusters a highly complex task.

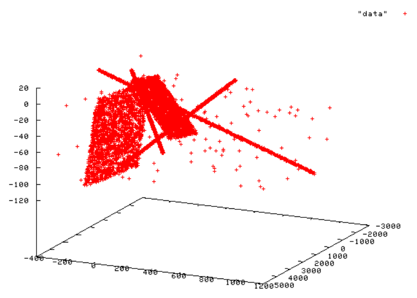
The predefined clusters in the synthetic data set have been separated clearly by COPAC (cf. Figure 9.8). As parameters we used $\mu = 50$ and $k = 50$. In order to determine a suitable value for ε we computed the $(\mu - 1)$-nearest neighbor distance diagram as suggested in Section 9.2.4. This way, we derived 0.004 as a suitable value for ε. Neither ORCLUS nor 4C were able to find the clusters equally well, although we test a broad range of parameter settings. Best results for ORCLUS were reported setting $l = 2$ and $k = 5$ (cf. Figure 9.9). For 4C we found the best results with $\varepsilon = 0.1$, $\mu = 50$, $\lambda = 2$, and $\delta = 0.25$ (cf. Figure 9.9).

In summary, COPAC shows better robustness against noise and parameter settings than ORCLUS and 4C. Due to the heuristics presented in Section 9.2.4, the parameter choice for COPAC was very easy and results in a complete detection of clusters. For both ORCLUS and 4C, we needed several runs with different parameters in order to produce the best but still not optimal (i.e. complete) results.

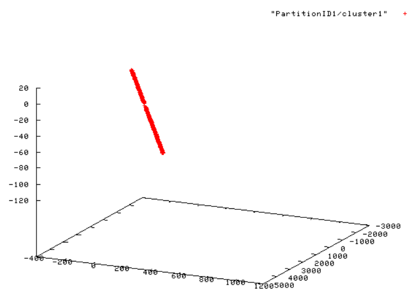
9.3.3 Results on Real-world Data

Metabolome data

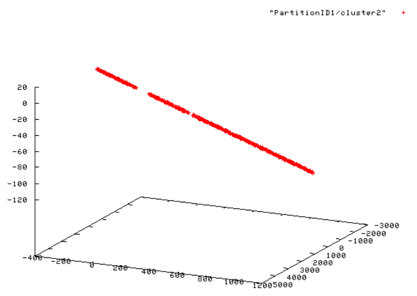
We used the Metabolome data set of [92] consisting of the concentrations of 43 metabolites in 20,391 human newborns. The newborns were labeled according to some specific metabolic diseases. The data contain 19,730 healthy newborns ("control"), 306 newborns suffering from phenylketonuria ("PKU"), and 355 newborns suffering from any other diseases ("other"). COPAC finds several pure or almost pure clusters. Table 9.1 shows the number of labeled newborns in each cluster, the remaining newborns were classified as noise. The parameters were chosen as suggested in Section 9.2.4 ($\varepsilon = 0.15$, $\mu = 10$, $k = 130 \approx 3 \cdot d$). ORCLUS never found clusters that were equally pure as the clusters found by COPAC although we tested a broad range of parameters. Using proper settings of the correlation dimensionality, 4C found some



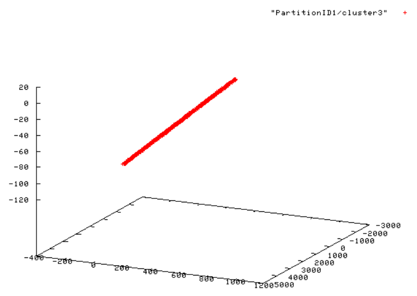
(a) Synthetic data set.



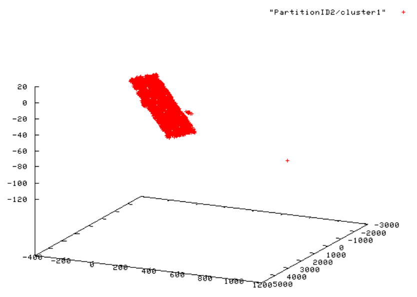
(b) Partition 1 – Cluster 1.



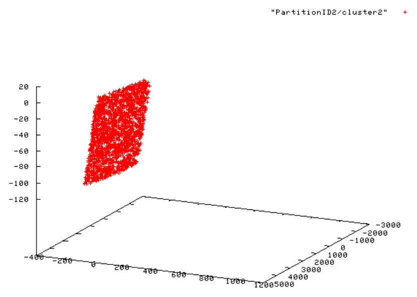
(c) Partition 1 – Cluster 2.



(d) Partition 1 – Cluster 3.

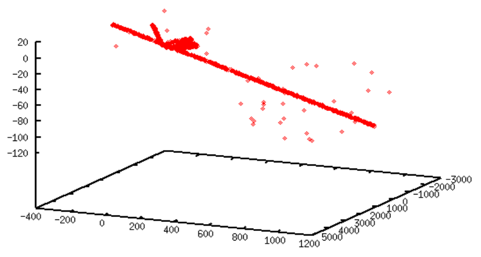


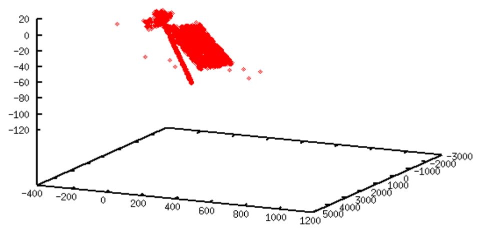
(e) Partition 2 – Cluster 1.

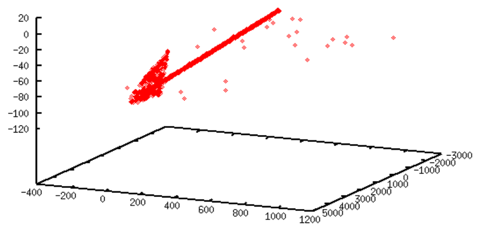


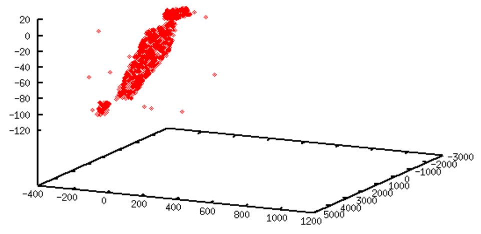
(f) Partition 2 – Cluster 2.

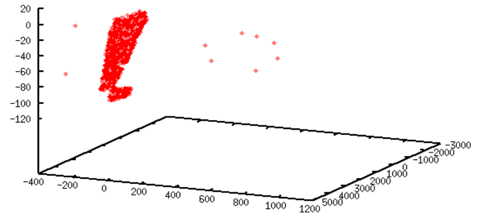
Figure 9.8: Synthetic data set: partitions and clustering with COPAC.

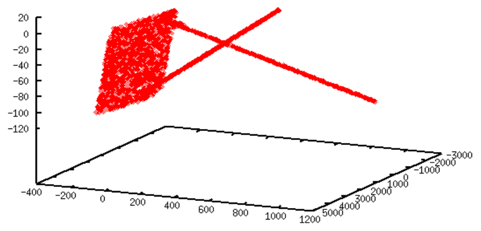

(a) ORCLUS – Cluster 1.


(b) ORCLUS – Cluster 2.


(c) ORCLUS – Cluster 3.


(d) ORCLUS – Cluster 4.


(e) ORCLUS – Cluster 5.


(f) 4C – Cluster 1.

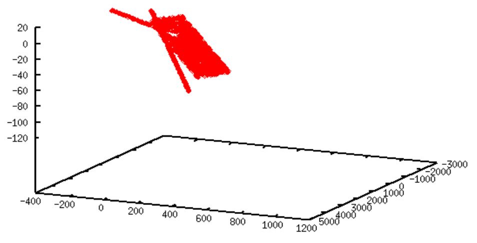

(g) 4C – Cluster 2.

Figure 9.9: Synthetic data set: clustering with ORCLUS and 4C.

Table 9.1: COPAC clustering on Metabolome data.

λ	c ID	# control	# PKU	# other
7	7-1	0	88	0
8	8-1	2	30	0
9	9-1	0	26	0
10	10-1	26	0	0
	10-2	0	41	3
11	11-1	0	27	32
	11-2	156	0	0
12	12-1	841	0	1
	12-2	0	4	33
13	13-1	2241	0	12
14	14-1	3411	2	23
15	15-1	5222	3	20
16	16-1	5561	3	8
17	17-1	1788	0	1
18	18-1	20	0	0
noise		460	82	222

pure “PKU” clusters, but were not able to detect pure “control” clusters. To derive information with 4C that is comparable to COPAC, several runs of 4C with different parameter settings are required, where each single run of 4C needs considerably more time than one complete run of COPAC.

Wages data

The Wages data set[2] 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. Parameters were chosen according to Section 9.2.4 ($\varepsilon = 0.01$, $\mu = 12$, $k = 12 = 3 \cdot d$), the results are summarized in Table 9.2. Since there are no predefined classes in this data set, models for the found clusters were derived using the algorithm as proposed in [4].

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

Table 9.2: COPAC clustering on Wages data.

c ID	λ	# objects	Description
1	2	188	YE = 12; A = *const*· YW
2	2	12	YE = 16; A = *const*· YW
3	3	98	YE + YW = A - 6
noise		236	

These models provide meaningful descriptions: 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. 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. Neither ORCLUS nor 4C were able to detect meaningful clusters in the wages data.

Wisconsin Breast Cancer data

The (original) Wisconsin Breast Cancer database [98] consists of 699 patients suffering from two types of breast cancer, benign ("B") and malignant ("M"). Each patient is represented by a 9-dimensional vector of specific biomedical features. Parameters were chosen according to Section 9.2.4 ($\varepsilon = 1.0$, $\mu = 30$, $k = 30 \approx 3 \cdot d$). COPAC detected six pure correlation clusters in this data set, the results are summarized in Table 9.3. Again COPAC outperforms its competitors since neither ORCLUS nor 4C were able to detect pure clusters in these data.

Gene expression data

This data set was derived from an experimental study of apoptosis in human tumor cells[3]. Apoptosis is a genetically controlled pathway of cell death. The

[3]The confidential data set is donated by our project partners.

Table 9.3: COPAC clustering on breast cancer data.

c ID	λ	# B	# M
1	2	108	0
2	3	12	0
3	4	100	0
4	5	46	0
5	5	0	113
6	6	0	126
noise		50	2

data set contains the expression level of 4610 genes at five different time slots (5, 10, 15, 30, 60 minutes) after initiating the apoptosis pathway. COPAC found two different, biologically relevant clusters of functionally related genes. Parameters were chosen according to Section 9.2.4 ($\varepsilon = 0.5$, $\mu = 10$, $k = 20$).

The first cluster contains about 20 genes related to the mitochondrion, especially to the mitochondrion membrane. The genes in the cluster exhibit a negative correlation of each of the first five time slots with the last time slot, i.e. the expression level decreases over time. This indicates that the volume of the energy metabolism (which is located in mitochondria) is decreasing during cell death. The second cluster that contains several genes that are related to the tumor necrosis factor (TNF) and several interleukin and interleukin receptor genes. Interleukins play a key role in the human immune system, especially in cancer response. The strong correlation with the TNF-related genes makes perfectly sense, since the cells respond to necrosis. The correlation among the genes in this cluster is similar to that of cluster 1 and, thus, also suggests that the activity of the corresponding genes decreases with proceeding cell death.

Table 9.4: COPAC clustering on expression data.

cID	sample gene names	description
1	NDUFB10, MTRF1, TIMM17A, CPS1, NM44, COX10, FIBP, TRAP1, MTERF, HK1, HADHA, ASAH2, CPS1, CA5A, BNI3PL, TOM34, ME2	proteins located in and/or related to mitochondrial membran
2	TNFRSF6, TNFRSF11A, TNFRSF7, TNFRSF1B, TNFRSF5, TNFRSF1A, TRAF5, TRAF2, TNFSF12	proteins related to tumor necrosis factor (TNF)
	IL1A, IL1B, IL2, IL6, IL10, IL18, IL24, IL1RN, IL2RG, IL4R, IL6R, IL7R, IL10RA, IL10RB, IL12A, IL12RB2, IL15RA, IL22R	interleukins or their receptors activating immune response

Chapter 10

Mining Hierarchies of Correlation Clusters: HiCO

In Part II it was demonstrated that a trivial combination of the concepts of clustering and correlation detection is not sufficient to find correlation clusters. Therefore, algorithms for correlation clustering have to integrate the concepts of clustering and correlation detection in a more sophisticated way. The algorithm ORCLUS [11], for instance, integrates PCA into k-means clustering and the algorithms 4C [35] and COPAC [6] integrate PCA into the density based clustering algorithm DBSCAN [47]. These algorithms decompose a data set into subsets of points, each subset being associated to a specific λ-dimensional hyperplane.

Since ORCLUS and 4C use the correlation dimensionality λ as a global parameter, i.e., the correlation dimensionality of the resulting clusters must be determined by the user, they are both not able to find all correlation clusters of different dimensionality during one single run. COPAC overcomes this drawback. However, searching clusters of different dimensionality is essentially a *hierarchical problem* because several correlation clusters of low dimensionality may together form a larger correlation cluster of higher dimensionality, and so on. For example, consider two lines in a 3D space that are embedded into a 2D plane (cf. Figure 10.1). Each of the two lines forms a 1-dimensional correlation cluster. On the other hand the plane is a

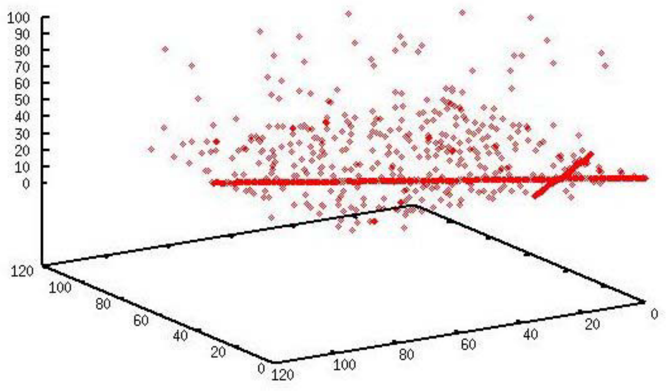


Figure 10.1: Hierarchies of correlation clusters.

2-dimensional correlation cluster that includes the two 1-dimensional correlation clusters. In order to detect the lines, one has to search for 1-dimensional correlation clusters, ($\lambda = 1$), whereas in order to detect the plane, one has to search for 2-dimensional correlation clusters ($\lambda = 2$).

None of the previously proposed algorithms for correlation clustering is able to detect hierarchies of correlation clusters. Therefore, in this chapter we propose HiCO (Hierarchical Correlation Ordering), a new algorithm for searching simultaneously for correlation clusters of arbitrary dimensionality and detecting hierarchies of correlation clusters. Additionally, HiCO overcomes another drawback of the existing non-hierarchical correlation clustering methods like 4C, COPAC and ORCLUS, since 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.

This chapter is organized as follows: Section 10.1 introduces basic definitions in a similar way as the previous chapters. However, the definitions vary slightly w.r.t. the definitions presented earlier as different concepts are based upon them. Section 10.2 introduces the algorithm HiCO with its main concepts and properties. An experimental evaluation in comparison to ORCLUS and 4C is presented in Section 10.3.

The concepts presented in this chapter have been published in [7].

10.1 Basic Definitions

To determine how two points are correlated, we introduce in Section 10.2 a special distance measurement called *correlation distance*. This measurement is based on the *local correlation dimensionality* of a point which reflects the dimensions having a strong correlation in the neighborhood of this point. In order to compute the correlation distance between two points we have to determine in a preprocessing step for each point P of the data set:

1. The *local covariance matrix* $\boldsymbol{\Sigma}_P$ which is the covariance matrix of the k nearest neighbors of P.
2. The *local correlation dimensionality* λ_P which indicates the dimensionality of the subspace accommodating the k nearest neighbors of P.
3. The *local correlation similarity matrix* $\hat{\boldsymbol{M}}_P$ which is used to compute the local correlation distance between two points.

In the following, $\mathcal{D}$ denotes a set of points and *dist* is the Euclidean distance function.

Definition 10.1 (local covariance matrix)
Let $k \in \mathbb{N}$, $k \leq |\mathcal{D}|$. The local covariance matrix $\boldsymbol{\Sigma}_P$ *of a point $P \in \mathcal{D}$ w.r.t. k is formed by the k nearest neighbors of P.*

Formally: Let $\overline{X}$ be the centroid of $NN_k(P)$, then

$$\boldsymbol{\Sigma}_P = \frac{1}{|NN_k(P)|} \cdot \sum_{X \in NN_k(P)} (X - \overline{X}) \cdot (X - \overline{X})^{\mathsf{T}}$$

Since the local covariance matrix $\boldsymbol{\Sigma}_P$ of a point P is a square matrix it can be decomposed into the eigenvalue matrix $\boldsymbol{E}_P$ *of P and the* eigenvector matrix $\boldsymbol{V}_P$ *of P such that $\boldsymbol{\Sigma}_P = \boldsymbol{V}_P \cdot \boldsymbol{E}_P \cdot \boldsymbol{V}_P^{\mathsf{T}}$.*

The eigenvalue matrix $\boldsymbol{E}_P$ is a diagonal matrix holding the eigenvalues of $\boldsymbol{\Sigma}_P$ in decreasing order in its diagonal elements. The eigenvector matrix $\boldsymbol{V}_P$ is an orthonormal matrix with the corresponding eigenvectors of $\boldsymbol{\Sigma}_P$.

Unlike in [35] we do not base the local covariance matrix on a range query predicate because for our hierarchical clustering method, we do not have a predefined query radius and there exists no natural choice for such a radius. Therefore, we prefer to base the local correlation on a certain number k of nearest neighbors which is more intuitive.

Now we define the local correlation dimensionality of a point P as the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes of the k nearest neighbors of P. If, for instance, the points in a certain region of the data space are located near by a common line, the correlation dimensionality of these points will be 1. That means we have to determine the principal components of the points in $NN_k(P)$. 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 $\boldsymbol{\Sigma}_P$ which is the total variance of the points in $NN_k(P)$. 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 k nearest 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 $\boldsymbol{E}_P$ for which*

$$\frac{\sum_{i=1}^{r} e_i}{\sum_{i=1}^{d} e_i} \geq \alpha$$

We call the first λ_p eigenvectors of $\boldsymbol{V}_P$ strong *eigenvectors, the remaining eigenvectors are called* weak*.*

Typically α is set to values between 0.8 and 0.9, e.g. $\alpha = 0.85$ denotes that the obtained principal components explain 85% of the total variance. In the following we denote the λ_P-dimensional subspace which is spanned by the major axes of the neighborhood of P the *correlation hyperplane* of P.

In the third step of the preprocessing phase we associate each point with a so-called local correlation similarity matrix which is used to compute the local correlation distance to another point of the data set. The local correlation similarity matrix can be derived from the local covariance matrix in the following way:

Definition 10.3 (local correlation similarity matrix)
Let point $P \in \mathcal{D}$, $\boldsymbol{V}_P$ the corresponding $d \times d$ eigenvector matrix of the local covariance matrix $\boldsymbol{\Sigma}_P$ of P, and λ_P the local correlation dimensionality of P. The matrix $\hat{\boldsymbol{E}}_P$ with diagonal entries $\hat{e}_i$ ($i = 1, \ldots, d$) is computed according to the following rule:

$$\hat{e}_i = \begin{cases} 0, & \text{if } i \leq \lambda_P \\ 1, & \text{otherwise} \end{cases}$$

The matrix $\hat{\boldsymbol{M}}_P = \boldsymbol{V}_P \cdot \hat{\boldsymbol{E}}_P \cdot \boldsymbol{V}_P^{\mathsf{T}}$ is called the local correlation similarity matrix *of P.*

Definition 10.4 (local correlation distance)
The local correlation distance *of point P to point Q according to the local correlation similarity matrix $\hat{\boldsymbol{M}}_P$ associated with point P is denoted by*

$$\text{LocDist}_\text{P}(P, Q) = \sqrt{(P - Q)^{\mathsf{T}} \cdot \hat{\boldsymbol{M}}_P \cdot (P - Q)}.$$

$\text{LocDist}_\text{P}(P, Q)$ is the weighted Euclidean distance between P and Q using the local correlation similarity matrix $\hat{\boldsymbol{M}}_P$ as weight. The motivation for the adaptation of $\hat{\boldsymbol{M}}_P$ is that the original local covariance matrix $\boldsymbol{\Sigma}_P$ 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 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 we need quite the opposite. (2) The eigenvalues vary with the data distribution, so some points P may have higher eigenvalues in $\boldsymbol{E}_P$ 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.

Intuitively spoken, the local correlation distance $\textsc{LocDist}_{\text{P}}(P, Q)$ equals the Euclidean distance between Q and the correlation hyperplane exhibited by the neighbors of P. Thus, if Q lies within the correlation hyperplane of P then $\textsc{LocDist}_{\text{P}}(P, Q) = 0$.

We call the matrix $\hat{\boldsymbol{E}}$ the *selection matrix of the weak eigenvectors* because $\boldsymbol{V}_P \cdot \hat{\boldsymbol{E}}$ provides a matrix containing only weak eigenvectors. We will later also use another matrix $\check{\boldsymbol{E}}_P = \boldsymbol{I}_{d \times d} - \hat{\boldsymbol{E}}_P$ where the 0 and 1-entries of the diagonal elements are changed. We call this matrix $\check{\boldsymbol{E}}_P$ the *selection matrix of the strong eigenvectors* since $\boldsymbol{V}_P \cdot \check{\boldsymbol{E}}_P$ provides a matrix containing only strong eigenvectors. The selection matrices $\hat{\boldsymbol{E}}_P$ and $\check{\boldsymbol{E}}_P$ only depend on the local correlation dimensionality λ_P: $\hat{\boldsymbol{E}}_P$ is a $d \times d$ diagonal matrix where the first λ_P diagonal elements are 0 and the remaining $d - \lambda_P$ diagonal elements are 1 (and *vice versa* for $\check{\boldsymbol{E}}_P$).

Note that the local correlation distance is not yet the similarity measure which is directly used in our hierarchical clustering algorithm. It is merely a construction element for the actual correlation distance measure which will be defined in the following section.

10.2 Hierarchical Correlation Clusters

Hierarchical clustering methods in general 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, for example, 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 [35] is to group those points of a data set into same clusters where the correlation is uniform. Our general idea is 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, our strategy is to merge those points into common clusters which have small correlation distances. A hierarchy of correlation clusters means that clusters with small correlation distances (e.g. lines) are nested in clusters with higher correlation distances (e.g. 2d-planes).

10.2.1 Main Concepts of HiCO

Once we have associated the points of our database with a local correlation dimensionality and with a decomposed local similarity matrix, we can now explain the main idea of our hierarchical clustering algorithm. Conventional hierarchical clustering algorithms like SLINK or OPTICS without the idea of correlation work as follows: They keep two separate sets of points, points which have already been placed in the cluster structure and those which have not. 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.

We will adapt this paradigm. In the context of hierarchical correlation clustering, the hierarchy is a containment hierarchy of the correlation primitives. Two or more correlation lines may together form a 2-dimensional correlation plane and so on. We simulate this behavior by defining a similarity measure between points which assigns a distance of 1, if these two points (together with their associated similarity matrices) share a common correlation line. If they share a common correlation plane, they have a distance of 2, etc. Sharing a common plane can mean different things: Both 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 we associate a pair of points with a distance measure with the properties mentioned before, we can generally use the well-known hierarchical

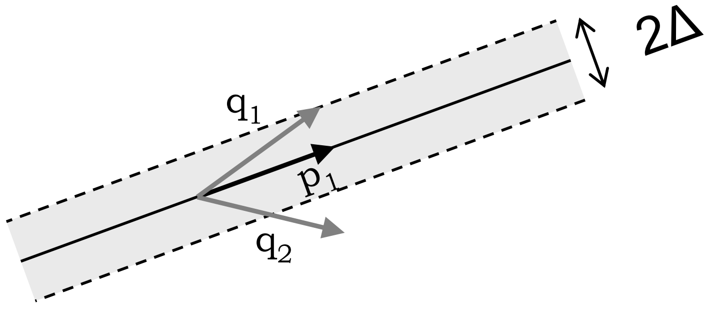


Figure 10.2: Spaces spanned by two vectors.

clustering algorithms. 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. By the notion *spanning a space* we 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 our context, a vector q adds a new dimension to the space spanned by a set of vectors $\{p_1, \ldots, p_n\}$ if the "difference" between q and the space spanned by $\{p_1, \ldots, p_n\}$ is substantial, i.e. if it exceeds the threshold parameter Δ. This is illustrated in Figure 10.2: the space spanned by $\{q_1\} \cup \{p_1\}$ is considered to be the same as the space spanned by p_1 only. On the other hand, the set of vectors $\{q_2\} \cup \{p_1\}$ span a 2-dimensional space, as the "difference" between q_2 and p_1 exceeds Δ.

We first give a definition of the *correlation dimensionality of a pair of points* $\lambda(P, Q)$ which follows the intuition of the spanned space. Later we will give a method for computing this dimensionality efficiently, given the local eigenvector matrices and the local correlation dimensionalities of P and Q, respectively. The correlation dimensionality is the most important component of our correlation similarity measure which will later be extended a little bit one more time.

Definition 10.5 (correlation dimensionality)
The correlation dimensionality *between 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 we would like to determine the correlation dimensionality of two points P and Q in a strong algebraic sense, we would have to look *exactly* for the linearly independent vectors in the union of the strong eigenvectors of P and Q. These n vectors form a basis of the n-dimensional subspace spanned by the strong eigenvectors of P and Q. Note that we are not interested in the spanned space in the strong algebraic sense. That means we are not looking for vectors that are linearly independent in strict algebraic sense but only for those vectors that are linearly independent in our 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 $p_i \in \boldsymbol{V}_P \cdot \check{\boldsymbol{E}}_P$ and $q_i \in \boldsymbol{V}_Q \cdot \check{\boldsymbol{E}}_Q$ in a one-by-one fashion. However, two vector pairs $\{p_1, p_2\}$ and $\{q_1, q_2\}$ can be linearly dependent although each vector is independent from each of the other three vectors.

We can test *one* of the strong eigenvectors, say $p_1 \in \boldsymbol{V}_P \cdot \check{\boldsymbol{E}}_P$ whether or not it is linearly independent (in our relaxed sense) to *all* the strong eigenvectors $q_i \in \boldsymbol{V}_Q \cdot \check{\boldsymbol{E}}_Q$ by substituting it into the local similarity matrix of Q, i.e. by testing: $p_1^\mathsf{T} \cdot \hat{\boldsymbol{M}}_Q \cdot p_1 > \Delta^2$.

If this comparison holds then we know that p_1 opens up a new dimension compared to Q, and that the correlation dimensionality $\lambda(P, Q)$ is at least $(\lambda_Q + 1)$. But if we test a second vector $p_2 \in \boldsymbol{V}_P \cdot \check{\boldsymbol{E}}_P$ we still have the problem that p_2 can be linearly dependent from $\boldsymbol{V}_Q \cdot \check{\boldsymbol{E}}_Q \bigcup \{p_1\}$ without being linearly dependent from any vector in $\boldsymbol{V}_Q \cdot \check{\boldsymbol{E}}_Q$ and $\{p_1\}$ alone. This problem is visualized in Figure 10.3. The strong eigenvectors p_1 and p_2 of P (depicted in dashed lines) are each linearly independent from the strong eigenvectors q_1 and q_2 of Q (depicted in solid lines) and by definition also linearly independent from each other (they are even orthogonal). However, p_2 is linearly dependent from the vectors in $\boldsymbol{V}_Q \cdot \check{\boldsymbol{E}}_Q \bigcup \{p_1\}$.

Therefore, before testing p_2, we have to integrate p_1 temporarily into the

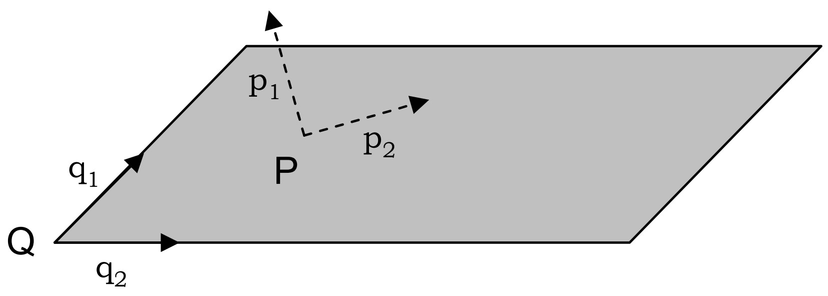


Figure 10.3: Two points with their eigenvectors.

eigenvector matrix $\boldsymbol{V}_Q$ (but only if p_1 indeed opens up a new dimension). To do so, we have to replace temporarily the *weak* eigenvector q_{λ_Q+1} by the new *strong* eigenvector p_1 and then orthonormalize the resulting matrix.

To orthonormalize the set of vectors $\{q_1, \ldots, q_{\lambda_Q}, p_1, q_{\lambda_Q+2}, \ldots, q_d\}$ the following steps have to be applied according to the method of Gram-Schmitt: Note that the vector q_{λ_Q+1} is temporarily replaced by vector p_1, i.e. $q_{\lambda_Q+1} = p_1$.

1. $x_i := q_i - \sum_{k=1}^{i-1} \langle q_k, q_i \rangle q_k$ for $i = \lambda_Q + 1, \ldots, d$
2. $q_i := \frac{x_i}{||x_i||}$ for $i = \lambda_Q + 1, \ldots, d$

The resulting vectors $\{q_1, \ldots, q_d\}$ build now again an orthonormal basis of the d-dimensional feature space.

We have to make $(d - \lambda_Q)$ vectors orthogonal which causes some problems because (1) we have to guarantee the linear independence (this time in the strong algebraic sense) of the remaining eigenvectors in $\boldsymbol{V}_Q$ because, otherwise, orthonormalization could fail. (2) The effort is considerable high because this orthonormalization (which is in $O(d^2)$) must be performed every time a new vector is integrated into $\boldsymbol{V}_Q$. Therefore, our actual algorithm computes the test $p_i^{\mathsf{T}} \cdot (\boldsymbol{V}_Q \cdot \hat{\boldsymbol{E}}_Q \cdot \boldsymbol{V}_Q^{\mathsf{T}}) \cdot p_i > \Delta^2$ in an indirect way by replacing $\hat{\boldsymbol{E}}$ by $\check{\boldsymbol{E}}$ 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 $\boldsymbol{V}$ be an orthonormal matrix consisting of the strong eigenvectors of $\boldsymbol{\Sigma}_Q$, some of the added and orthonormalized eigenvectors of $\boldsymbol{\Sigma}_P$ and the remaining orthonormalized weak eigenvectors of $\boldsymbol{\Sigma}_Q$. Then

$$x^{\mathsf{T}} \cdot (\boldsymbol{V} \cdot \hat{\boldsymbol{E}} \cdot \boldsymbol{V}^{\mathsf{T}}) \cdot x \;=\; x^{\mathsf{T}} \cdot x - x^{\mathsf{T}} \cdot (\boldsymbol{V} \cdot \check{\boldsymbol{E}} \cdot \boldsymbol{V}^{\mathsf{T}}) \cdot x$$

Proof.

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

□

The advantage of this computation is that now in the joint matrix $\check{\boldsymbol{M}}_Q = \boldsymbol{V}_Q \cdot \check{\boldsymbol{E}}_Q \cdot \boldsymbol{V}_Q^{\mathsf{T}}$ the weak eigenvectors q_m for $m > \lambda_Q$ are not considered. Keeping the weak eigenvectors orthonormal after every insertion of a new strong eigenvector of $\boldsymbol{\Sigma}_P$ causes the main effort in orthonormalization: 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 $\boldsymbol{V}_Q$ is given, because this vector is even linearly independent in our relaxed sense (and linear independency in weak sense implies linear independency in strict sense).

The algorithm for computing the correlation distance is presented in Figures 10.4 and 10.5. First, the correlation dimensionality $\lambda(P, Q)$ for a pair of points (P, Q) is derived as follows: For each of the *strong* eigenvectors q_i of Q test whether $q_i^{\mathsf{T}} \cdot q_i - q_i^{\mathsf{T}} \cdot (\boldsymbol{V}_P \cdot \check{\boldsymbol{E}}_P \cdot \boldsymbol{V}_P^{\mathsf{T}}) \cdot q_i > \Delta^2$. If so, increase λ_P by one and set p_{λ_P} to the orthonormalized vector of q_i. Finally, $\lambda_P(Q)$ contains the correlation dimensionality of the point pair (P, Q) w.r.t. P. In an analogue way $\lambda_Q(P)$ is derived for the same point pair. The overall correlation dimensionality $\lambda(P, Q)$ is the maximum of both, $\lambda_P(Q)$, and $\lambda_Q(P)$. $\lambda(P, Q)$ is now the major building block for our 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. We resolve these tie situations 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 as by a conventional hierarchical clustering method. Formally we define:

Definition 10.6 (correlation distance)
The correlation distance *between two points* $P, Q \in \mathcal{D}$*, denoted by* $\textsc{CDist}(P,Q)$*, is a pair consisting of the correlation dimensionality of P and Q and the Euclidean distance between P and Q, i.e.* $\textsc{CDist}(P,Q) = (\lambda(P,Q), dist(P,Q))$.

We say $\textsc{CDist}(P,Q) \leq \textsc{CDist}(R,S)$ *if one of the following conditions holds:*

(1) $\lambda(P,Q) < \lambda(R,S)$,

(2) $\lambda(P,Q) = \lambda(R,S)$ *and* $dist(P,Q) \leq dist(R,S)$.

10.2.2 Algorithm HiCO

Using the correlation distance defined above as a distance measure, we can basically run every hierarchical (or even non-hierarchical) clustering algorithm which is based on distance comparisons. Examples for such algorithms are Single-Link, Complete-Link, and the density-based clustering methods DBSCAN (non-hierarchical) and OPTICS. Since OPTICS is hierarchical and more robust w.r.t. noise than Single- and Complete-Link, we use the algorithmic schema and visualization technique of OPTICS for HiCO.

As suggested in [16] we introduce a smoothing factor μ to avoid the Single-Link effect and to achieve robustness against noise points. Thus, instead of using the correlation distance $\textsc{CDist}(P,Q)$ to measure the similarity of two points P and Q we use the *correlation reachability* $\textsc{Reach}_{\mu}(O,P)$ to compare these two points. The correlation reachability of a point P relative from a

```
**function correlationDistance**($P$, $Q$, $\Delta$)
  compute $\check{\boldsymbol{E}}_P$ from $\hat{\boldsymbol{E}}_P$;
  $\boldsymbol{V}_P$ = eigenvector matrix of $P$;
  $\lambda_P$ = correlation dimensionality of $P$;
  compute $\check{\boldsymbol{E}}_Q$ from $\hat{\boldsymbol{E}}_Q$;
  $\boldsymbol{V}_Q$ = eigenvector matrix of $Q$;
  $\lambda_Q$ = correlation dimensionality of $Q$;
  **for each** strong eigenvector $q_i \in \boldsymbol{V}_Q$ **do**
    **if** $q_i^{\mathsf{T}} q_i - q_i^{\mathsf{T}} \boldsymbol{V}_P \check{\boldsymbol{E}}_P \boldsymbol{V}_P^{\mathsf{T}} q_i > \Delta^2$ **then**
      adjust( $\boldsymbol{V}_P$, $\check{\boldsymbol{E}}_P$, $q_i$, $\lambda_P$);
      $\lambda_P = \lambda_P + 1$;
    **end if**
  **end for**
  **for each** strong eigenvector $p_i \in \boldsymbol{V}_P$ **do**
    **if** $p_i^{\mathsf{T}} p_i - p_i^{\mathsf{T}} \boldsymbol{V}_Q \check{\boldsymbol{E}}_Q \boldsymbol{V}_Q^{\mathsf{T}} p_i > \Delta^2$ **then**
      adjust( $\boldsymbol{V}_Q$, $\check{\boldsymbol{E}}_Q$, $p_i$, $\lambda_Q$);
      $\lambda_Q = \lambda_Q + 1$;
    **end if**
  **end for**
  $\text{CDist} = \max(\lambda_P, \lambda_Q)$;
  **return** $(\text{CDist}, dist_{Euclid}(P, Q))$;
**end**
```

Figure 10.4: Pseudo code correlation distance.

```
procedure adjust($\boldsymbol{V}$, $\check{\boldsymbol{E}}$, $x$, $\lambda$)
  // set column $(\lambda + 1)$ of matrix $V$ to vector $x$
  $v_{\lambda+1} := x$;
  for each strong eigenvector $v_i \in \boldsymbol{V}$ do
    $v_{\lambda+1} := v_{\lambda+1} - \langle v_i, x \rangle \cdot v_i$
  $v_{\lambda+1} := \frac{v_{\lambda+1}}{||v_{\lambda+1}||}$;
  set column $(\lambda$+$1)$ of $\check{\boldsymbol{E}}$ to the $(\lambda$+$1)$-th unit vector;
end
```

Figure 10.5: Pseudo code orthonormalization.

```
algorithm HiCO($\mathcal{D}$, $k$, $\mu$, $\alpha$, $\Delta$)
  //1. Preprocessing
  for each $P \in \mathcal{D}$ do
    compute $\hat{\boldsymbol{E}}_P$, $\boldsymbol{V}_P$;
  end for
  //2. Clustering
  // priority queue $pq$ is ordered by $\textsc{Reach}_\mu$
  for each $P \in \mathcal{D}$ do
    $P.\textsc{Reach} = \infty$;
    insert $P$ into $pq$;
  end for
  while $pq \neq \emptyset$ do
    $O := pq$.next();
    $R := \mu$-nearest neighbor if $O$;
    for each $P \in pq$ do
      new_cr $:= \max(\textsc{CDist}(O, R), \textsc{CDist}(O, P))$;
      $pq$.update($P$, new_cr);
    end for
  end while
end
```

Figure 10.6: Pseudo code HiCO algorithm.

point O is defined as the maximum value of the correlation distance from O to its μ-nearest neighbor and the correlation distance between P and O.

Definition 10.7 (correlation reachability)
For $\mu \in \mathbb{N}$, $\mu \leq |\mathcal{D}|$ let R be the μ-nearest neighbor of $O \in \mathcal{D}$ w.r.t. the correlation distance. The correlation reachability *of a point $P \in \mathcal{D}$ relative from point O w.r.t. $\mu \in \mathbb{N}$ is defined as*

$$\textsc{Reach}_{\mu}(O, P) = \max(\text{C}\textsc{Dist}(O, R), \text{C}\textsc{Dist}(O, P))$$

Using this correlation reachability, HiCO computes a “walk” through the data set and assigns to each point O its smallest correlation reachability with respect to a point considered before O in the walk. In each step of the algorithm HiCO selects that point O having the minimum correlation reachability to any already processed point. This process is managed by a seed list which stores all points that have been reached from already processed points sorted according to the minimum correlation reachabilities. A special order of the database according to its correlation-based clustering structure is generated, the so-called cluster order, which can be displayed in a correlation reachability diagram. Such a correlation reachability diagram consists of the 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 clustering structure of the data set which is very easy to understand. The “valleys” in the plot represent the clusters, since points within a cluster typically have lower correlation reachabilities than points outside a cluster.

The complete integration of our distance measure into the algorithm HiCO can be seen in Figure 10.6.

10.2.3 Runtime Complexity

Let n be the number of data points and d be the dimensionality of the data space. In the preprocessing step the correlation neighborhoods are precomputed which requires $O(nd + kd^2)$ for the determination of the covariance

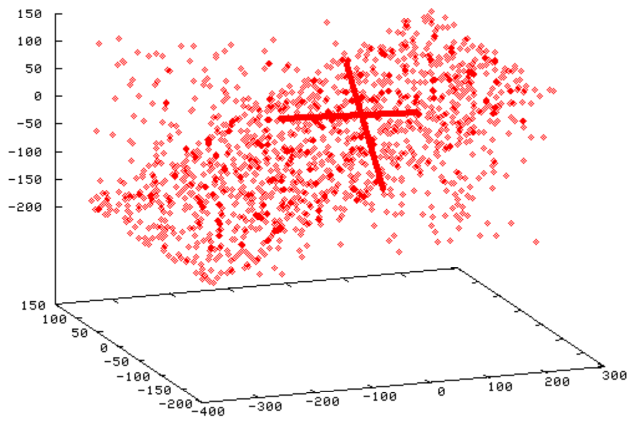


Figure 10.7: 3D synthetic data set (DS1).

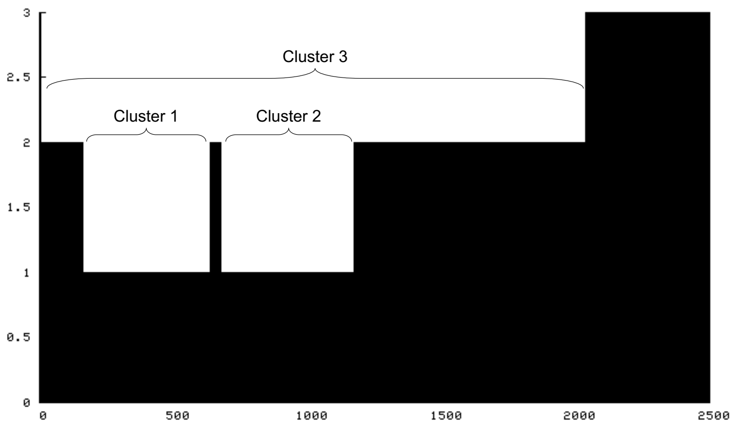
matrix. Since this is done for each object in the data set and $k < n$, we have a runtime complexity of $O(n^2d^2)$ for the preprocessing step. During the run of HiCO, we have to evaluate for each pair of points of the database its correlation dimensionality. This requires again a complexity of $O(n^2d^2)$. In addition, we have to decompose the covariance matrix of each point into the eigenvalue matrix and the eigenvector matrix. This step has a complexity of $O(nd^3)$. Thus, the overall runtime complexity of HiCO is $O(n^2d^2 + nd^3)$.

10.3 Evaluation

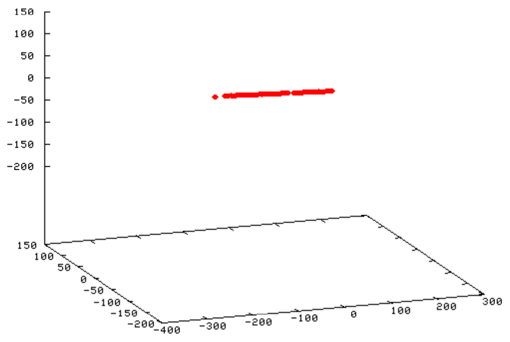
10.3.1 Data Sets

For our experiments, we used several synthetic data sets containing points marked with cluster labels that represent the hierarchical clustering structure. In addition, we used four real-world data sets. The first one, called "DS2", is a data set 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 suffer from the disease and on 50 carriers of the disease.

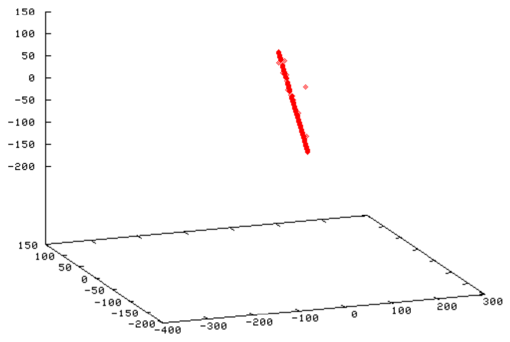
As a second data set we used the "El Nino" data, a benchmark data set



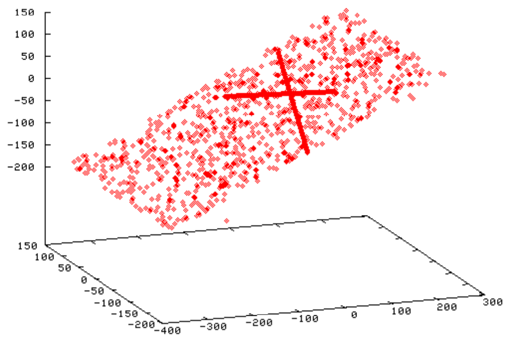
(a) Reachability plot.



(b) Cluster 1.

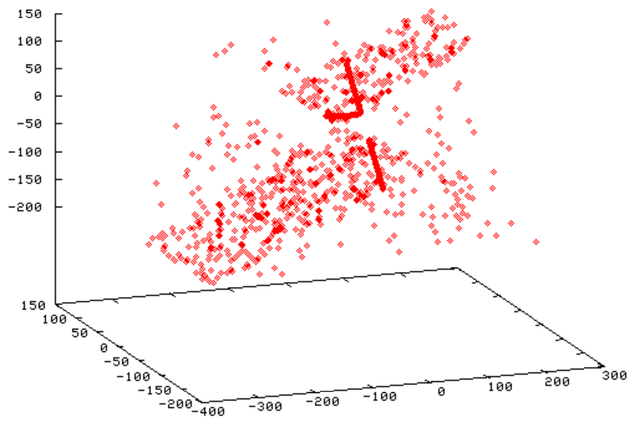


(c) Cluster 2.

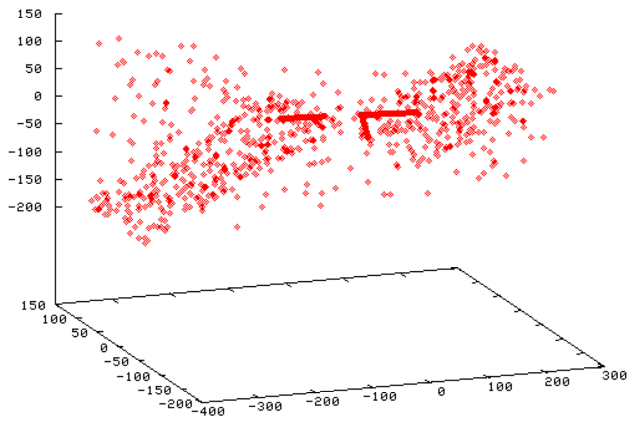


(d) Cluster 3.

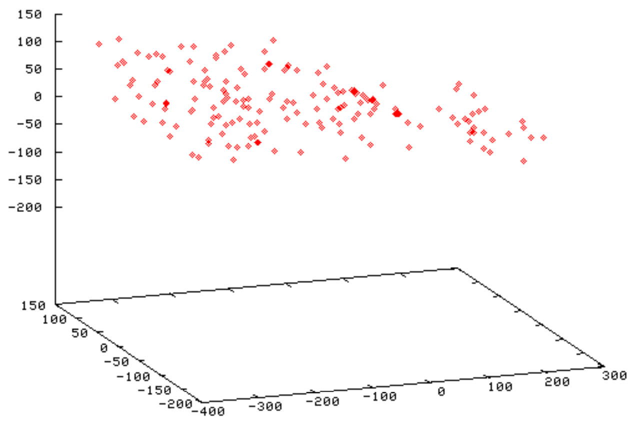
Figure 10.8: Results of HiCO applied to DS1 (Parameters: $\mu = k = 20$, $\alpha = 0.90$, $\Delta = 0.05$).



(a) Cluster 1.

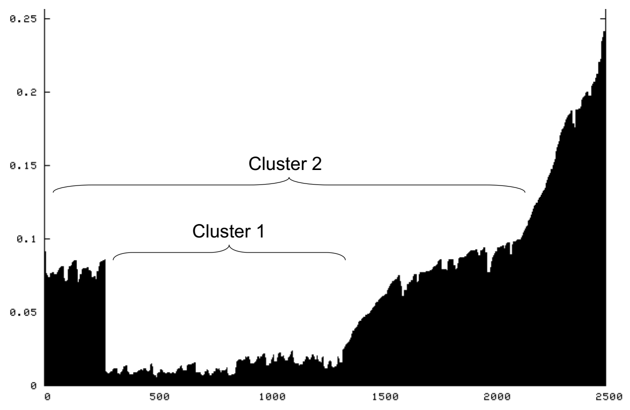


(b) Cluster 2.

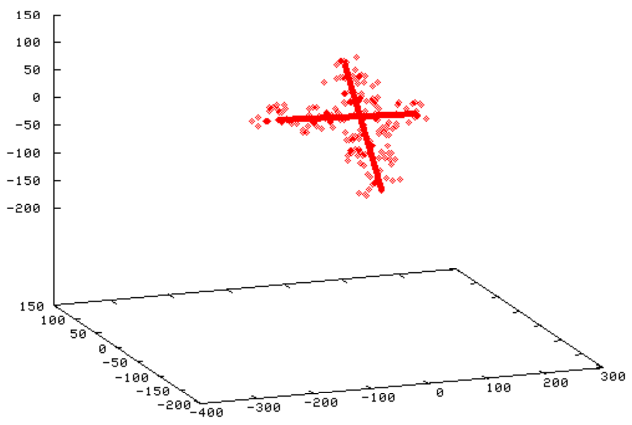


(c) Cluster 3.

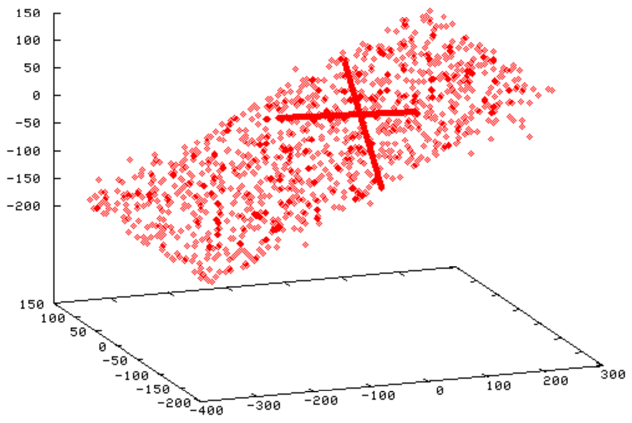
Figure 10.9: Results of ORCLUS applied to DS1 (Parameters: $k = 3$, $l = 2$).



(a) Reachability plot.



(b) Cluster 1.



(c) Cluster 2.

Figure 10.10: Results of OPTICS applied to DS1 (Parameters: $\varepsilon = 1$, $minPts = 20$).

from the UCI KDD Archive[1]. The data set called "DS3" contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. It contains approximately 800 9D objects. The third data set called "DS4" consists of approximately 550 11D observations from the 1985 Current Population Survey[2]. The fourth data set called "DS5" consists of the concentrations of 43 metabolites in 2,000 newborns. The newborns were labeled according to some specific metabolic diseases.

10.3.2 Results on Synthetic Data

We applied HiCO to several synthetic data sets. In the following, we focus on the 3-dimensional data set "DS1" depicted in Figure 10.7. It contains a hierarchy of correlation clusters consisting of two 1D correlations (lines) belonging to a 2D correlation (plane) and noise. The correlation reachability distance diagram computed by HiCO is shown in Figure 10.8(a). As it can be observed, HiCO detects two 1D correlation clusters that are embedded within a 2D correlation cluster. Additionally, some objects have a correlation distance of 3 (which equals the data dimensionality), i.e. they can be regarded as noise. We analyzed the "valleys" in the correlation reachability diagram marked with "Cluster 1", "Cluster 2", and "Cluster 3". The points that are clustered together in that correlation clusters are depicted in Figures 10.8(b), 10.8(c), and 10.8(d). As it can be seen, the correlation plane "Cluster 3" corresponds to the 2D correlation cluster in the diagram, whereas the two correlation lines "Cluster 1" and "Cluster 2" exactly correspond to the 1D correlation sub-clusters of "Cluster 3" in the diagram. Obviously, HiCO detects the hidden correlation hierarchy exactly.

For comparison, we applied ORCLUS, OPTICS and 4C on the same data sets, but none of them were able to find the correlation clusters equally well, despite reasonable parameter settings. For ORCLUS we choose e.g. $k = 3$ and $l = 2$, but as it can be seen in Figure 10.9, ORCLUS was not able to

[1]`http://kdd.ics.uci.edu/`

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

find the correlation clusters hidden in the synthetic 3D data set.

We also applied OPTICS to the synthetic 3D data set (cf. Figure 10.10). OPTICS detected a hierarchy of clusters according to its density based paradigm, but it was not able to separate the correlation within these clusters.

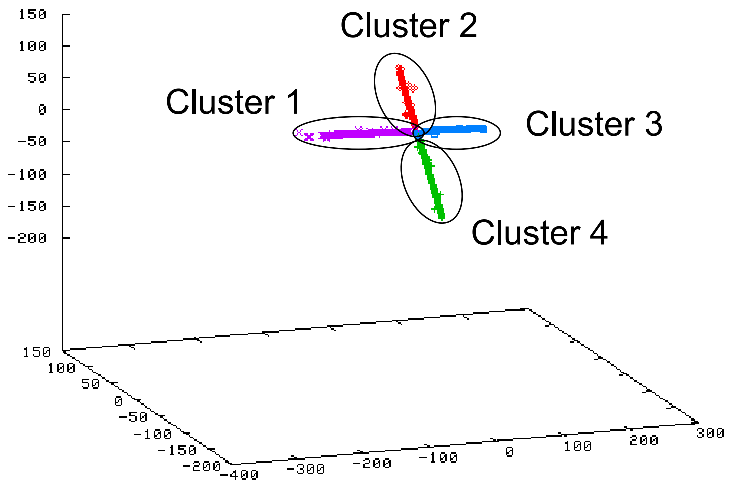
Figure 10.11 shows the results of two 4C runs with different parameter settings. As parameter λ was set to one in the first run, 4C detected four 1-dimensional clusters consisting of the two lines in the synthetic 3D data set (cf. Figure 10.11(a)), but 4C failed to detect the 2-dimensional correlations. According to the parameter setting of $\lambda = 2$ in the second run, 4C found one 2-dimensional correlation cluster consisting of the two lines and the plane (cf. Figure 10.11(b)). In both runs, 4C was not able to detect all three correlation clusters as HiCO did.

10.3.3 Real-world Data

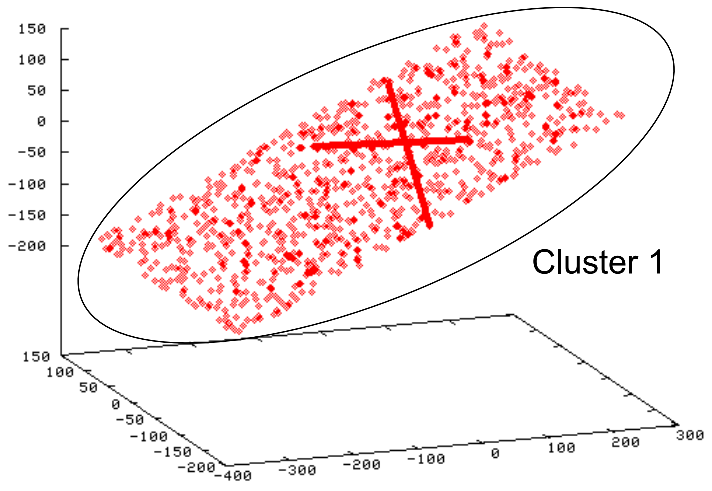
The results of HiCO applied to the real-world data sets are shown in Figure 10.12. Applied to the DS2 data (cf. Figure 10.12(a)), HiCO found a cluster with correlation dimensionality of 2 embedded in a larger 3-dimensional correlation cluster. The cluster with a correlation dimensionality of 2 mostly 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.

The resulting reachability diagram of HiCO applied on data set DS3 is depicted in Figure 10.12(b). As it can be seen, the hierarchy contains a 1-dimensional correlation cluster and four 2-dimensional clusters. Analyzing these clusters, we found that the observations belonging to these clusters were mostly made from neighbored buoys.

The result of HiCO on DS4 is depicted in Figure 10.12(c). We can observe a strong hierarchy of correlation clusters. HiCO computed four 2-dimensional correlation clusters embedded in a 3-dimensional correlation cluster which is again embedded in a 4-dimensional cluster. The hierarchy ends up with 5-dimensional and 6-dimensional clusters. The first of the 2-dimensional clus-

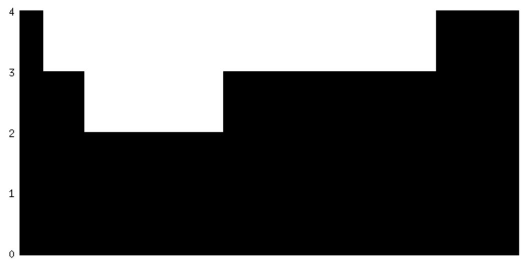


(a) Parameters: λ=1, ε=0.1, μ=20, δ=0.2.

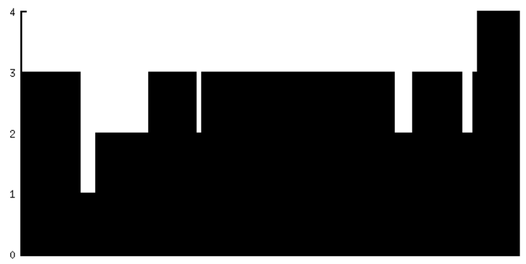


(b) Parameters: λ=2, ε=0.25, μ=20, δ=0.1.

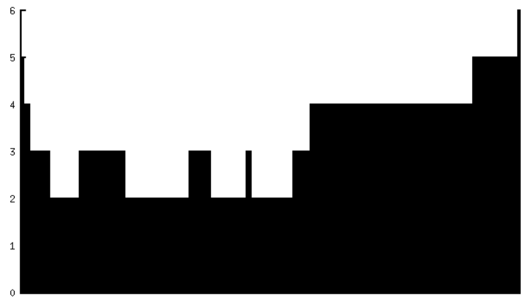
Figure 10.11: Results of 4C applied to DS1.



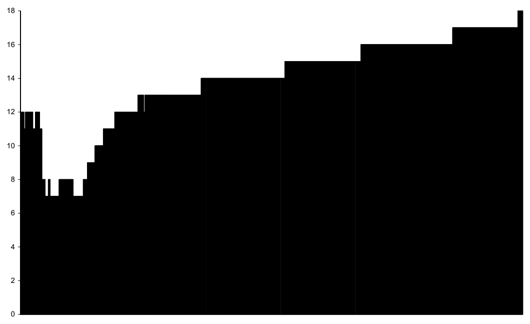
(a) DS2 (μ=10, k=25).



(b) DS3 (μ=15, k=40).



(c) DS4 (μ=10, k=40).



(d) DS5 (μ=10, k=100).

Figure 10.12: Results of HiCO on real-world data sets (Parameters: $\Delta = 0.25$, $\alpha = 0.8$).

ters consists only of 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 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.

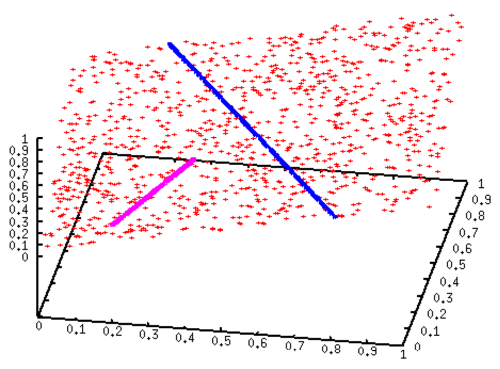
Finally, HiCO retrieved on DS5 7-dimensional and 8-dimensional correlation clusters embedded in higher dimensional clusters (cf. Figure 10.12(d)). These clusters of relative low dimensionality consist only of newborns suffering from phenylketonuria (PKU), while the healthy newborns are grouped in the clusters of higher dimensionality.

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

Chapter 11

Exploring Complex Hierarchical Relationships of Correlation Clusters: ERiC

The first approach that can detect correlation clusters is ORCLUS [11], integrating PCA into k-means clustering. The algorithm 4C [35] and its variant COPAC [6] integrate PCA into a density-based clustering algorithm. These approaches can be seen as "flat" approaches in the following sense. A correlation cluster C_1 with dimensionality λ_1 may be embedded in (and therefore may be part of) another correlation cluster C_2 with dimensionality $\lambda_2 > \lambda_1$. In general, there may be a kind of hierarchy among correlation clusters that are embedded into higher dimensional correlation clusters. Since neither ORCLUS nor 4C and COPAC can detect such hierarchies, the algorithm HiCO [7] was proposed tackling correlation clustering as a hierarchical problem, i.e. exploring the information of correlation clusters of lower correlation dimensionality that together form a larger correlation cluster of higher correlation dimensionality. Although it is represented by the same models (dendrogram or reachability diagram), the resulting hierarchy is different from the hierarchies computed by traditional hierarchical clustering algorithms such as Single-Link or OPTICS [16]. The hierarchy among correlation clusters reflects the relationships among the subspaces in which these correlation


(a) Sample data set 1

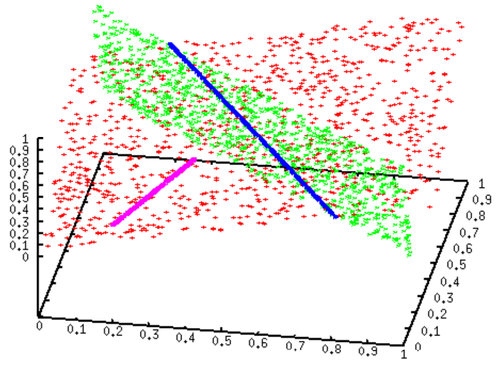

(b) Sample data set 2

Figure 11.1: Simple (a) and complex (b) hierarchical relationships among correlation clusters

clusters are embedded rather than spatial vicinity or density. As a simple illustration consider the data set depicted in Figure 11.1(a): Two lines, i.e. 1-D correlation clusters, are embedded within a plane, i.e. a 2-D correlation cluster. The resulting hierarchy consists of the two 1-D clusters as leaf-nodes of the hierarchy-tree both having the 2-D correlation cluster as parent node. HiCO aims at generating a tree-based representation of the correlation cluster hierarchy.

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). Consider e.g. the data set depicted in Figure 11.1(b): One of the 1-D correlation clusters is the intersection of two 2-D correlation clusters, i.e. it is embedded within two clusters of higher dimensionality. Those multiple inclusions can only be represented by a graph-based visualization approach which is beyond the capabilities of previous methods such as HiCO.

In this chapter, we propose a new algorithm called ERiC (Exploring Relationships among Correlation clusters) to completely uncover any 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.

This chapter is organized as follows: We first elaborate in more detail the drawbacks of the predecessor method HiCO (Section 11.1). We recall the notion of correlation clusters formally in Section 11.2. Section 11.3 presents the algorithm ERiC to overcome the shortcomings of HiCO. An experimental evaluation of ERiC in comparison to ORCLUS, 4C, and HiCO is shown in Section 11.4.

The material presented in this chapter has been partially published in [5].

11.1 Motivation: Drawbacks of HiCO

HiCO incorporates a distance measure taking into account local correlation dimensionalities into the hierarchical clustering algorithm OPTICS [16]. The resulting reachability-plot allows to derive a simple hierarchy of correlation clusters. Let us consider two main drawbacks of HiCO: Firstly, HiCO uses a relatively complex distance measure for every distance query in the clustering step. This results in considerable computational efforts. Secondly, HiCO assumes a relatively simple hierarchy of correlation clusters. Multiple inclusions cannot be derived from the resulting plot. Thus, the detected hierarchical structure of correlation clusters can be misleading or even simply wrong.

This limitation is illustrated in Figure 11.2 depicting the resulting reachability plot when applying HiCO on the sample datasets from Figure 11.1. As it can be observed, the resulting plots look almost identical for both, sample data set 1 (cf. Figure 11.2(a)) and sample data set 2 (cf. Figure 11.2(b)). Since valleys in the plot indicate clusters, both plots reveal the same information of two 1-D clusters embedded within one 2-D cluster. In fact, in data set 2 the two 2-D clusters cannot be separated and the complex hierarchy consisting of the multiple inclusion cannot be detected by HiCO. The true

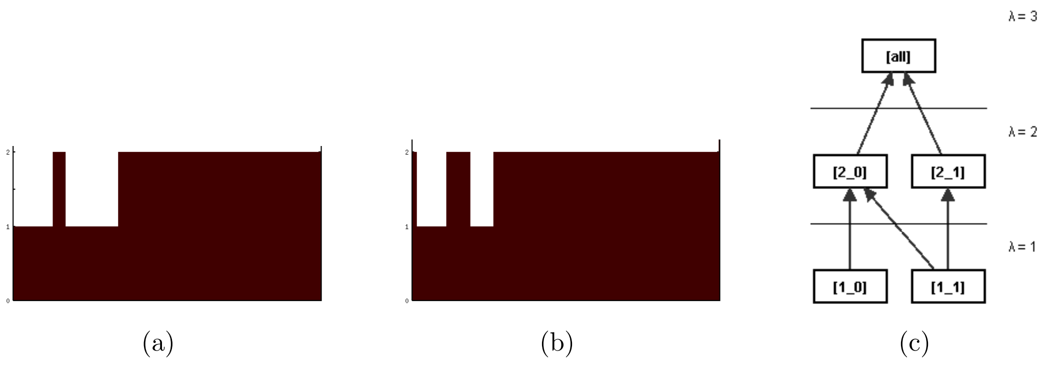


Figure 11.2: (a) and (b): Results of HiCO on the data sets shown in Figure 11.1 and (c): the true hierarchical relationships

hierarchy hidden in sample data set 2 can only be represented by a graph model. Figure 11.2(c) envisions such a visualization of the complete hierarchy allowing for multiple inclusions. In fact, our method ERiC will produce such a visualization.

11.2 A Notion of Correlation Clusters

In this section, we prepare the introduction of our approach by formalizing the notion of correlation clusters. In the following, we assume $\mathcal{D}$ to be a database of n feature vectors in a d-dimensional feature space, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$. A correlation cluster is a set of feature vectors that are close to a common, arbitrarily oriented subspace of a given dimensionality λ ($1 \leq \lambda < d$). In the data space, the correlation cluster appears as a hyperplane of dimensionality λ.

In general, one way to formalize the concept of correlation clusters is to use PCA. Formally, let $\mathcal{C}$ be a correlation cluster, i.e. $\mathcal{C} \subseteq \mathcal{D}$, and let $\bar{X}$ denote the centroid of all points in $\mathcal{C}$. The $d \times d$ *covariance matrix* $\boldsymbol{\Sigma}_\mathcal{C}$ of $\mathcal{C}$ is defined as:

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

Since the covariance matrix $\boldsymbol{\Sigma}_\mathcal{C}$ of $\mathcal{C}$ is a positive semi-definite square matrix, it can be decomposed into the *eigenvalue matrix* $\boldsymbol{E}_\mathcal{C}$ of $\boldsymbol{\Sigma}_\mathcal{C}$ and the *eigenvector matrix* $\boldsymbol{V}_\mathcal{C}$ of $\boldsymbol{\Sigma}_\mathcal{C}$ such that $\boldsymbol{\Sigma}_\mathcal{C} = \boldsymbol{V}_\mathcal{C} \cdot \boldsymbol{E}_\mathcal{C} \cdot \boldsymbol{V}_\mathcal{C}^\mathsf{T}$. The eigenvalue matrix $\boldsymbol{E}_\mathcal{C}$ is a diagonal matrix storing the d non-negative eigenvalues of $\boldsymbol{\Sigma}_\mathcal{C}$ in decreasing order. The eigenvector matrix $\boldsymbol{V}_\mathcal{C}$ is an orthonormal matrix with the corresponding d eigenvectors of $\boldsymbol{\Sigma}_\mathcal{C}$.

Now we define the correlation dimensionality of $\mathcal{C}$ as the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes in $\boldsymbol{V}_\mathcal{C}$. Let us 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 by a common line, the correlation dimensionality of these points will be 1. That means we have to determine the principal components (eigenvectors) of $\boldsymbol{\Sigma}_\mathcal{C}$. 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 $\boldsymbol{\Sigma}_\mathcal{C}$ 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 $\alpha \in]0,1[$ of the total variance of $\mathcal{C}$.

In the following, we call the $\lambda_\mathcal{C}$-dimensional subspace which is spanned by the major axes of $\mathcal{C}$ the *correlation hyperplane* of $\mathcal{C}$. Since we follow the convention that the eigenvalues are ordered decreasingly in the eigenvalue matrix $\boldsymbol{E}_\mathcal{C}$, the major axes correspond to the $\lambda_\mathcal{C}$ first eigenvectors of $\boldsymbol{\Sigma}_\mathcal{C}$.

Thus, the correlation dimensionality $\lambda_\mathcal{C}$ is the dimensionality of the subspace containing all points of the set $\mathcal{C}$ allowing a small deviation corresponding to the remaining portion of variance of $1 - \alpha$. The remaining, neglected variance scatters along the eigenvectors $v_{\lambda_\mathcal{C}+1}, \ldots, v_d$.

11.3 Algorithm ERiC

As discussed above, hierarchical clustering schemata such as the agglomerative schema (e.g. used by Single-Link), the divisive schema, or the density-based schema (e.g. used by OPTICS) cannot uncover complex hierarchies that exhibit multiple inclusions. The reason for this 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, 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 to first generate all correlation clusters and, second, to determine the hierarchy from this result. Obviously, during the computation of the clusters it would be very helpful to aggregate information that can be used to explore the hierarchical relationships among these clusters. In addition, it is required to compute all correlation clusters for all possible correlation dimensions simultaneously.

Since none of the existing correlation clustering algorithms meets both requirements we propose a novel approach to determine the complete set of correlation clusters and additional information for the hierarchy generation process. In particular, our algorithm ERiC consists of the following three steps: First, the objects of the database are partitioned w.r.t. their "correlation dimensionality" (cf. Section 11.3.1) in a similar way as proposed for the algorithm COPAC (cf. Chapter 9). This correlation dimensionality of a point $p \in \mathcal{D}$ will reflect the dimensionality of the correlation cluster in which p fits best. In a second step, the points within each partition are clustered using a "flat" correlation clustering algorithm (cf. Section 11.3.2). The result of these two steps is the complete set of correlation clusters with the additional information regarding their dimensionality. To explore the relationships among the correlation clusters found during step 2, we follow a bottom-up strategy. For any cluster $\mathcal{C}_i$ with correlation dimensionality λ_i, we consider those clusters $\mathcal{C}_j$ with correlation dimensionality $\lambda_j > \lambda_i$ 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 11.3.3).

11.3.1 Partitioning w.r.t. Correlation Dimensionality

In the first step of ERiC, we partition the database according to the *local correlation dimensionality* of the database objects reflecting the correlation dimensionality of the local neighborhood of each point.

Definition 11.1 (local correlation dimensionality)
Let $\alpha \in]0,1[$, $p \in \mathcal{D}$, and let $\mathcal{N}_p$ denote the set of points in the local neighborhood of p. Then the local correlation dimensionality λ_p *of the point p is the smallest number of eigenvalues e_i in the eigenvalue matrix $\boldsymbol{E}_{\mathcal{N}_p}$ explaining a portion of at least α of the total variance, i.e.*

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

Let us note that $\boldsymbol{E}_{\mathcal{N}_p}$ is the eigenvalue matrix of $\boldsymbol{\Sigma}_{\mathcal{N}_p}$ which is the covariance matrix of $\mathcal{N}_p$. 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. The set of points $\mathcal{N}_p$ of p should well reflect the correlation in the local neighborhood of p. Thus, one may e.g. choose the k-nearest neighbors of p as the neighborhood $\mathcal{N}_p$ of p. This way, one can ensure to consider a set of points large enough to derive a meaningful covariance matrix $\boldsymbol{\Sigma}_{\mathcal{N}_p}$. 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.

Based on Definition 11.1, the first step of ERiC partitions the database objects according to their local correlation dimensionality, derived from the k-nearest neighbors of each object. A point $p \in \mathcal{D}$ with $\lambda_p = i$ is 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. In terms of correlation clustering, $\mathcal{D}_d$ contains noise, since there is no linear dependency of features within the neighborhood of p, if $\lambda_p = d$.

This first step of ERiC yields an appropriate correlation dimensionality for each point in advance. Furthermore, the number n of data points to process in the clustering step for each partition is reduced to $\frac{n}{d}$ on the average.

11.3.2 Computing Correlation Clusters within each Partition

Having performed the partitioning of the database $\mathcal{D}$ in step 1, a clustering step is performed for each partition separately. For the clustering procedure, we can utilize the fact that all points within a given partition $\mathcal{D}_i$ share a common local correlation dimensionality i. This enables a much more efficient procedure in comparison to HiCO. Based on the local correlation dimensionality of a point p, we distinguish *strong eigenvectors* that span the hyperplane associated with a possible correlation cluster containing p, and *weak eigenvectors* that are perpendicular to this hyperplane.

Definition 11.2 (strong and weak eigenvectors)
Let $p \in \mathcal{D}$, λ_p be the local correlation dimensionality of p, and let $\boldsymbol{V}_p$ be the corresponding eigenvectors of the point p based on the local neighborhood $\mathcal{N}_p$ of p. We call the first λ_p eigenvectors of $\boldsymbol{V}_p$ strong eigenvectors*, the remaining eigenvectors are called* weak*.*

To easily describe some matrix computations in the following, we define a selection matrix for weak eigenvectors as follows.

Definition 11.3 (selection matrix for weak eigenvectors)
Let $p \in \mathcal{D}$, λ_p be the local correlation dimensionality of p, and let $\boldsymbol{E}_p$ be the corresponding eigenvectors and eigenvalues of the point p based on the local neighborhood $\mathcal{N}_p$ of p. The selection matrix $\hat{\boldsymbol{E}}_p$ for weak eigenvectors

with diagonal entries $\hat{e}_i \in \{0,1\}$, $i = 1, \ldots, d$, is constructed according to the following rule:

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

Based on this definition, the *weak eigenvectors* of p are given by $\boldsymbol{V}_p \cdot \hat{\boldsymbol{E}}_p$.

For the clustering, we will associate two points, $p, q \in \mathcal{D}_i$, to the same 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*. The number i of strong eigenvectors is the same for p and q as both are placed in the same partition $\mathcal{D}_i$. But we can even define this condition more general for a different number of strong eigenvectors. However, we need to consider linear dependency in a weakened sense to allow a certain degree, say Δ, of deviation. In real world data, it is unlikely to find a correlation cluster that perfectly fits to a hyperplane. We therefore define an *approximate linear dependency* among the strong eigenvectors of two points.

Definition 11.4 (approximate linear dependency)
Let $\Delta \in]0,1[$, $p, q \in \mathcal{D}$, and w.l.o.g. $\lambda_p \le \lambda_q$. Then the strong eigenvectors of p are approximately linear dependent *from the strong eigenvectors of q if the following condition holds for all strong eigenvectors v_i of p:*

$$\sqrt{v_i^{\mathsf{T}} \cdot \boldsymbol{V}_q \cdot \hat{\boldsymbol{E}}_q \cdot \boldsymbol{V}_q^{\mathsf{T}} \cdot v_i} \le \Delta$$

If the strong eigenvectors of p are approximately linear dependent *from the strong eigenvectors of q, we write*

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

As indicated above, Δ specifies the degree of deviation of a straight plane a correlation cluster may exhibit.

Definition 11.4 does not take into account any affinity. Thus, we consider the strong eigenvectors of p approximately linear dependent from the strong eigenvectors of q ($\textsc{span}(p) \subseteq^{\Delta}_{\text{aff}} \textsc{span}(q)$), although possibly $p \notin \textsc{span}(q)$, i.e., the space spanned by the strong eigenvectors of p is (approximately) parallel to the space spanned by the strong eigenvectors of q. To exclude affine subspaces, we additionally assess the distance between p and q along the weak eigenvectors of q, i.e., perpendicular to the hyperplane defined by the strong eigenvectors of q. This distance which we call *affine distance* is defined as follows.

Definition 11.5 (affine distance)
Let $p, q \in \mathcal{D}$, w.l.o.g. $\lambda_p \leq \lambda_q$, and $\textsc{span}(p) \subseteq^{\Delta}_{\text{aff}} \textsc{span}(q)$. The affine distance *between p and q is given by*

$$\textsc{dist}_{\text{aff}}(p, q) = \sqrt{(p - q)^{\mathsf{T}} \cdot \boldsymbol{V}_q \cdot \hat{\boldsymbol{E}}_q \cdot \boldsymbol{V}_q^{\mathsf{T}} \cdot (p - q)}$$

Combining approximate linear dependency (Definition 11.4) and the affine distance (Definition 11.5) yields the definition of a correlation distance between two points.

Definition 11.6 (correlation distance)
Let $\delta \in \mathbb{R}^+_0$, $\Delta \in]0,1[$, $p, q \in \mathcal{D}$, and w.l.o.g. $\lambda_p \leq \lambda_q$. Then the correlation distance CDIST *between two points $p, q \in \mathcal{D}$, denoted by* CDIST(p, q)*, is defined as follows*

$$\text{CD}\textsc{ist}(p, q) = \begin{cases} 0 & \textit{if } \textsc{span}(p) \subseteq^{\Delta}_{\text{aff}} \textsc{span}(q) \\ & \quad \wedge \textsc{dist}_{\text{aff}}(p, q) \leq \delta \\ 1 & \textit{otherwise} \end{cases}$$

The parameter δ specifies a certain degree of jitter. Two parallel subspaces M, N are considered distinct, if the affine distances $\textsc{dist}_{\text{aff}}(m, n)$ and $\textsc{dist}_{\text{aff}}(n, m)$ exceed δ for any two points $m \in M$ and $n \in N$, otherwise the subspaces are considered to be equal. Since the relations $\textsc{span}(p) \subseteq^{\Delta}_{\text{aff}} \textsc{span}(q)$ and $\textsc{dist}_{\text{aff}}(p, q)$ are based on the local neighborhood of q, they are not symmetric. For $\lambda_p < \lambda_q$, these measurements yield the notion of a subspace $\textsc{span}(p)$ embedded in another subspace $\textsc{span}(q)$ of higher dimensionality as required to deduct a hierarchy of subspaces. However, as a distance function for clustering within one partition, all clusters are supposed to exhibit equal dimensionality. We therefore construct a symmetric distance function as

$$dist(p, q) = \max\left(\text{CD}\textsc{ist}(p, q), \text{CD}\textsc{ist}(q, p)\right).$$

In each partition, we perform a density-based clustering using DBSCAN with *dist* as distance function. DBSCAN is chosen due to its efficiency, effectiveness, and usability: The input parameter ε determining the range of neighborhood is set to 0 since the distance d is binary. The parameter μ (= *MinPts* in DBSCAN) determines the minimum size of a cluster. This parameter can be intuitively set according to the nature of a given problem. As a result, we get a set of clusters for each partition $\mathcal{D}_i$.

11.3.3 Aggregating the Hierarchy of Correlation Clusters

As mentioned above, the parent of a cluster $\mathcal{C}_i$ with correlation dimensionality λ_i can be any cluster $\mathcal{C}_j$ with correlation dimensionality $\lambda_j > \lambda_i$. Each cluster $\mathcal{C}_i$ derived in step 2 gets assigned a centroid x_i as mean value over all cluster members. Then the cluster centroid x_i gets assigned a set of strong and weak eigenvectors using all cluster members as neighborhood $\mathcal{N}_{x_i}$ as specified in Definitions 11.1 and 11.3. Assuming the clusters sorted in ascending order w.r.t. their correlation dimensionality (as already given by the partitions $\mathcal{D}_1, \ldots, \mathcal{D}_d$), ERiC starts with the first cluster $\mathcal{C}_m$ and checks for each cluster $\mathcal{C}_n$ with $\lambda_n > \lambda_m$ whether $\textsc{span}(x_m) \subseteq^{\Delta}_{\text{aff}} \textsc{span}(x_n)$ and $\textsc{dist}_{\text{aff}}(x_m, x_n) \leq \delta$ according to Definitions 11.4–11.5, i.e. the $\text{CD}\textsc{ist}(x_m, x_n)$ is derived (Definition 11.6). If $\text{CD}\textsc{ist}(x_m, x_n) = 0$, cluster $\mathcal{C}_n$ is treated as parent of cluster $\mathcal{C}_m$, unless $\mathcal{C}_n$ is a parent of any cluster $\mathcal{C}_o$ that in turn is already a parent of $\mathcal{C}_m$, because in that case the relationship between $\mathcal{C}_n$ and $\mathcal{C}_m$ is that of a grandparent. The pseudo code of this procedure is depicted in Figure 11.3.

The resulting hierarchy is visualized using a graph-like representation. An example is depicted in Figure 11.2(c) showing the hierarchy of correlation

```
**method** buildHierarchy(ClusterList $cl$)
    // $cl = \langle C_i \rangle$ *is sorted w.r.t.* $\lambda_{C_i}$
    $\lambda_{max} := d$; // $d$ = *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** $\text{CDist}(\mathcal{C}_i, \mathcal{C}_j) = 0 \wedge$
                  ($\mathcal{C}_i$.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 11.3: The method to build the hierarchy of correlation clusters.

```
**method** isParent(Cluster $\mathcal{P}$, ClusterList $cl$)
    **for each** $\mathcal{C} \in cl$ **do**
        **if** $\text{CDist}(\mathcal{P}, \mathcal{C}) = 0$ **then**
            **return true**;
        **end if**
    **end for**
    **return false**;
**end.**
```

Figure 11.4: The method to check wether a cluster is parent of one of the clusters in a list.

clusters in sample data set 2 (cf. Figure 11.1). 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 graph contains all objects in partition $\mathcal{D}_d$. All correlation clusters with equal correlation dimensionality are placed at the same level below the root. Thus, 1D 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 indicates a (containment) relationship. In fact, a node N represents a cluster of all objects assigned to N as well as all objects assigned to child nodes of N.

11.3.4 Runtime Complexity

The preprocessing step of ERiC works for each point as follows: First a k-nearest neighbor query is performed, which has a complexity of $O(n)$ 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 using PCA which requires $O(d^3)$ time. Thus, for all points together we have a time complexity of $O(n^2 + k \cdot d^2 \cdot n)$ in the first step of ERiC, since $d \ll k$ as discussed above.

Applying DBSCAN to the data set in the second step of ERiC results in a time complexity of $O(d^3 \cdot n_i^2)$, where n_i is the number of points in partition i. This is due to the fact, that the original DBSCAN has a worst case complexity of $O(n^2)$ on top of the sequential scan. Applying the correlation distance as given in Definition 11.6, the overall time complexity of DBSCAN is $O(d^3 \cdot n_i^2)$. Assuming on average a uniform distribution of the points over all possible correlation dimensionalities, all partitions contain $\frac{n}{d}$ points. Thus, for d partitions, the required runtime reduces to $O(d^2 \cdot n^2)$.

The hierarchy aggregation considers all pairs of clusters $(\mathcal{C}_i, \mathcal{C}_j)$ associated to different partitions (i.e., $\lambda_i < \lambda_j$) and determines the CDist for the corresponding cluster representatives. Let $|\mathcal{C}|$ be the number of clusters.

Then the complexity of this method corresponds to $O(|\mathcal{C}|^2 \cdot d^3)$. However, in the experimental evaluation, we show the hierarchy aggregation to require only a marginal runtime compared to the overall runtime of ERiC. This is due to the fact that $|\mathcal{C}| \ll n$.

Thus, the overall runtime complexity of ERiC can be considered as $O(n^2 \cdot d^2)$.

11.4 Evaluation

11.4.1 Effectiveness

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 11.5) contains 3-dimensional objects grouped in a complex hierarchy of four 1-dimensional and three 2-dimensional correlation clusters with a multiple inclusion and some noise points. The results of ERiC applied to "DS1" using a parameter setting of $k = 16, \mu = 30, \alpha = 0.85, \delta = \Delta = 0.1$ are shown in Figure 11.6. In the upper left Figure 11.6(a) the three 2-dimensional correlation clusters found by ERiC are marked with different colors, the upper right Figure 11.6(b) shows the four 1-dimensional correlation clusters found by ERiC. In the lower Figure 11.6(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

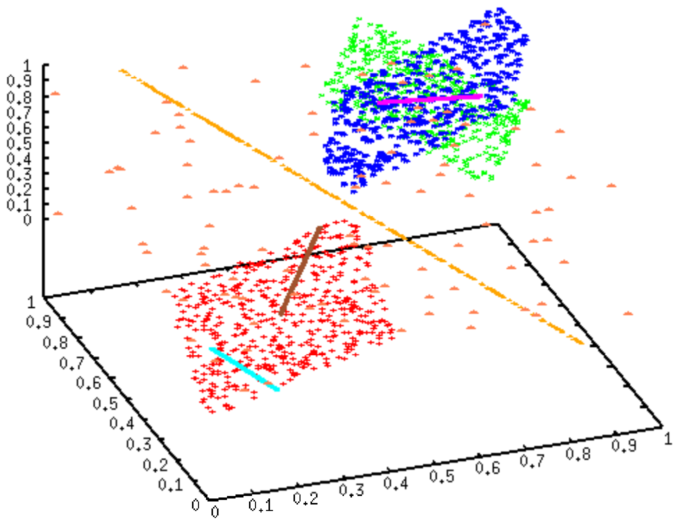
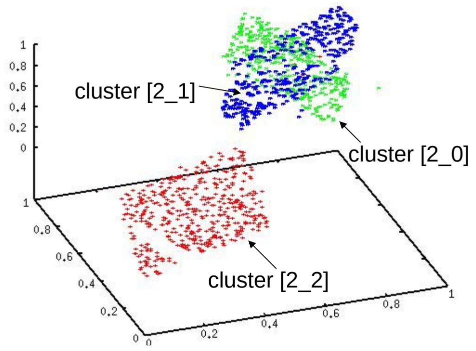
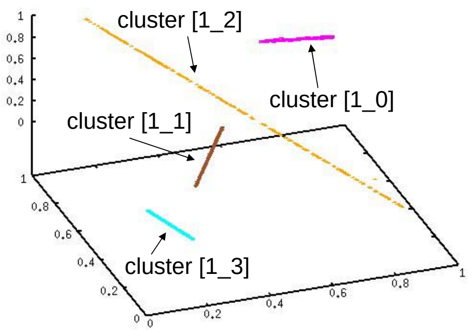


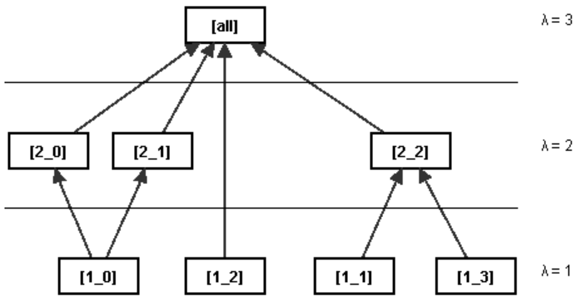
Figure 11.5: Data set "DS1".



(a) 2-dimensional correlation clusters.

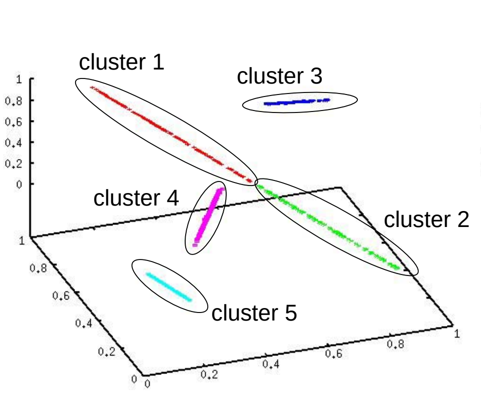


(b) 1-dimensional correlation clusters.

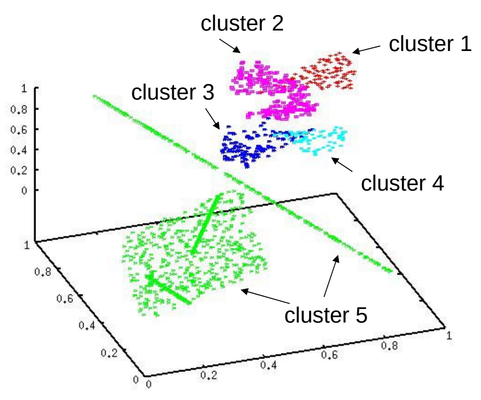


(c) Correlation clustering graph.

Figure 11.6: Results of ERiC on "DS1".



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

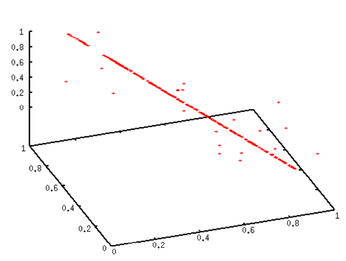


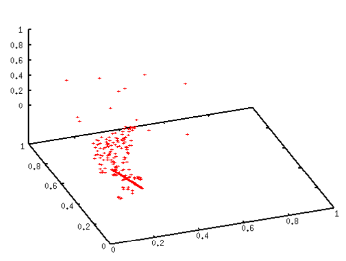
(b) Results of 4C with $\lambda = 2$.

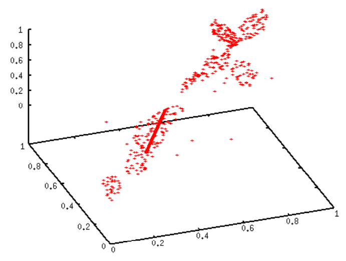
Figure 11.7: Results of 4C with different λ-parameter settings on "DS1".

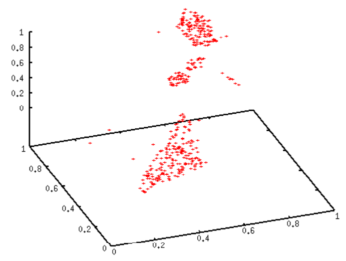
"1_2"), and a multiple inclusion of one 1-dimensional cluster (line "1_0") embedded within two 2-dimensional clusters (planes "2_1" and "2_2").

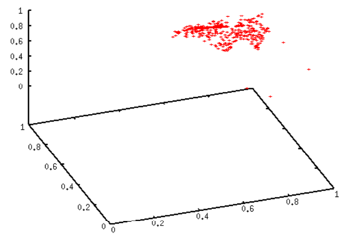
For comparison, ORCLUS, 4C, and HiCO have also been applied to data set "DS1", but none of the existing state-of-the-art correlation clustering approaches performs equally well. The algorithm 4C can produce a "flat" clustering, i.e., 4C can either detect the 1-dimensional correlation clusters or the 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 11.7. The left Figure 11.7(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 11.6) into two clusters. The three 2-dimensional planes have been classified as noise in this run. In the right Figure 11.7(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 11.6). 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 11.8, one can see that ORCLUS

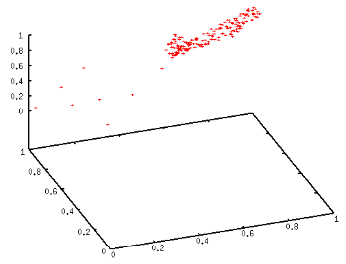

(a) ORCLUS - cluster 1.


(b) ORCLUS - cluster 2.


(c) ORCLUS - cluster 3.


(d) ORCLUS - cluster 4.


(e) ORCLUS - cluster 5.


(f) ORCLUS - cluster 6.

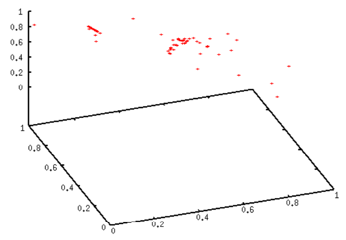

(g) ORCLUS - cluster 7.

Figure 11.8: Results of ORCLUS on "DS1".

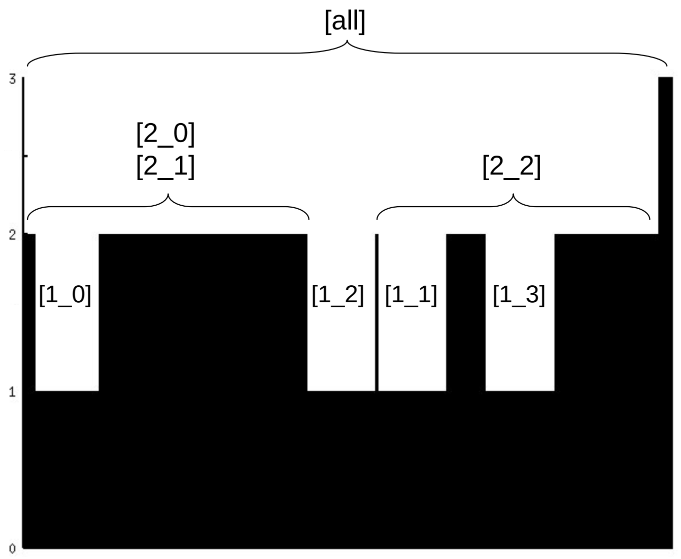


Figure 11.9: Result of HiCO on "DS1".

completely failed to detect all correlation clusters in data set "DS1".

Since both 4C and ORCLUS produce a flat clustering, no hierarchy can 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 11.9. 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 competitors all fail to produce the true clusters and the proper hierarchy.

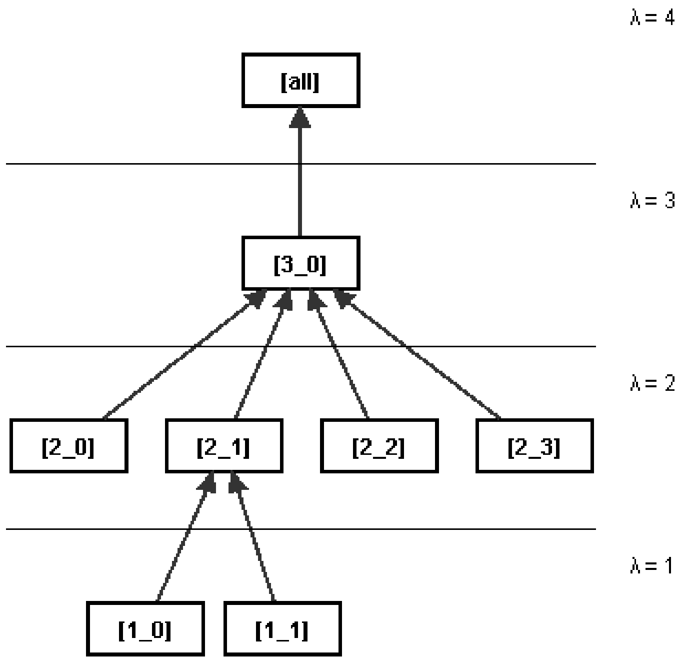
Real-world Data Sets

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 $k = 5, \mu = 4, \alpha = 0.85, \delta = \Delta = 0.01$. The results are shown in Figure 11.10(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 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 11.10(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 $k = 30, \mu = 30, \alpha = 0.85, \delta = \Delta = 0.75$ is depicted in Figure 11.11. 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

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

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



(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 11.10: Results of ERiC on "Wages" data.

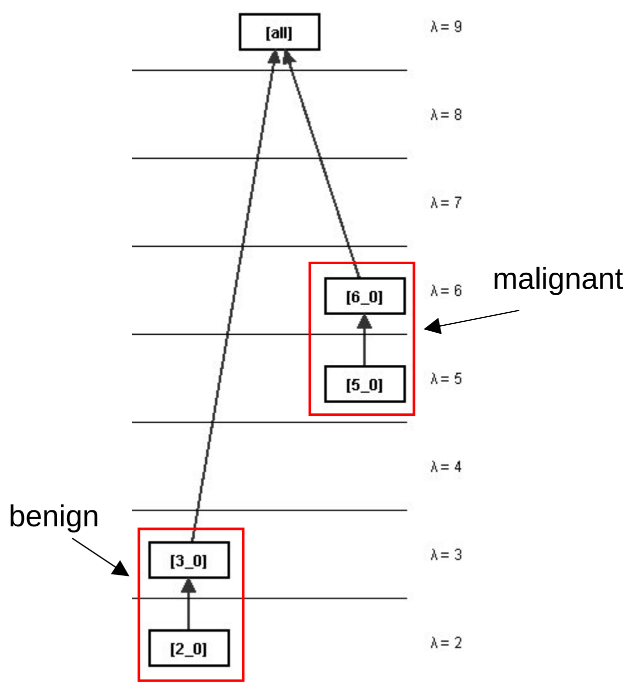


Figure 11.11: Results of ERiC on "Breast Cancer" data.

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 these 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 $k = 15, \mu = 10, \alpha = 0.85, \delta = \Delta = 0.5$ is depicted in Figure 11.12. 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.

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

[3]`http://www.ics.uci.edu/~mlearn/databases/pendigits/`

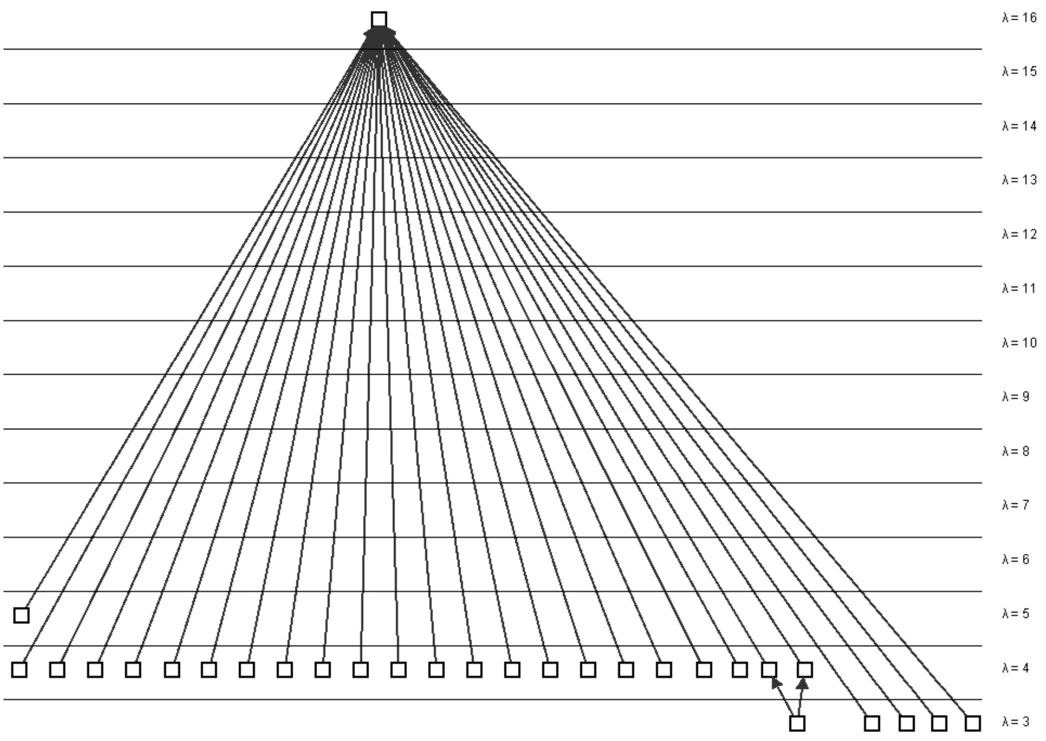


Figure 11.12: Results of ERiC on “Pendigits” data.

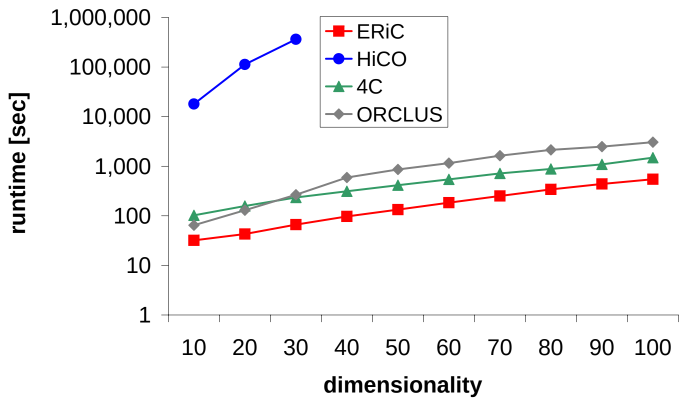


Figure 11.13: Runtime of ERiC, HiCO, 4C, and ORCLUS w.r.t. the dimensionality.

11.4.2 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 $k = 50, \mu = 500, \alpha = 0.999, \delta = \Delta = 0.01$. HiCO has been applied to the data sets with a parameter setting of $k = 50, \mu = 500, \alpha = 0.999, \Delta = 0.01$. 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 11.13, ERiC clearly outperforms the other competitors (please note the logarithmic scale of the runtime-axis).

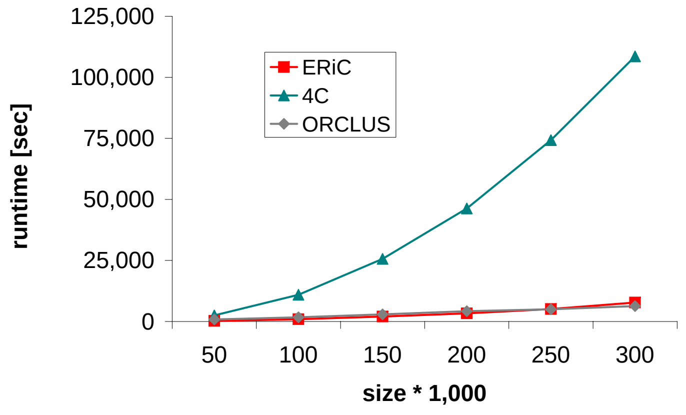


Figure 11.14: Runtime of ERiC, 4C, and ORCLUS w.r.t. the size of the data set.

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 $k = 50, \mu = 2,500, \alpha = 0.999, \delta = \Delta = 0.01$. 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 11.14 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

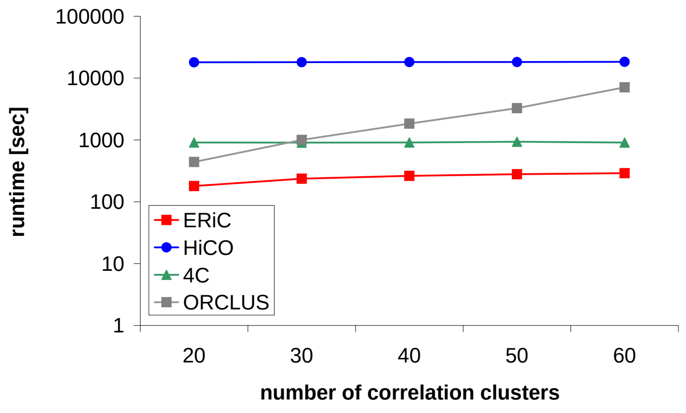


Figure 11.15: Runtime of ERiC, HiCO, 4C and ORCLUS w.r.t. the number of clusters.

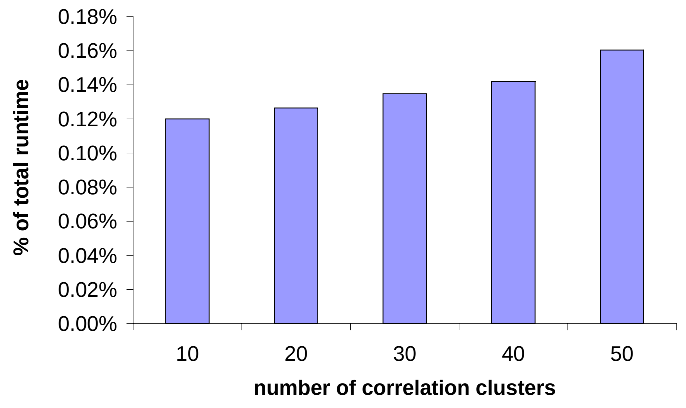


Figure 11.16: Runtime of the third step of ERiC (hierarchy aggregation) w.r.t. the number of clusters.

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 11.15 the runtime of ERiC is quite robust w.r.t. the number of correlation clusters in the data set (and also the runtimes of HiCO and 4C are). 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 11.16 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 11.3.4, 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.

Chapter 12

Increasing the Robustness of PCA-based Correlation Clustering Algorithms

The major challenge of correlation clustering is identifying the correct subspace of a cluster. Most correlation clustering algorithms [11, 35, 136, 7, 6, 5] apply principal component analysis (PCA) to a subset of points in order to define the correct subspace in orientation and weighting of the transformed axes. PCA is a mature technique and allows the construction of a broad range of similarity measures grasping local correlation of attributes and, therefore, allows to find arbitrarily oriented subspace clusters. It is easy to see that the more points of this subset are cluster members that are located on the common hyperplane, the more accurate the procedure of determining the correct subspace (i.e. hyperplane) will be. However, a drawback common to all those approaches is the notorious *locality assumption.* Since cluster memberships of points are obviously not known beforehand, it is assumed that the local neighborhood, e.g. the ε-neighborhood or the k-nearest neighbors, of cluster points or cluster centers represents the correct subspace suitably well in its orientation and variance along axes. This assumption is widely accepted but it boldly contradicts the basic problem statement, i.e. "find clusters in a high dimensional space", because high dimensional spaces are typically doomed by the *curse of dimensionality.* The term “curse of dimensionality” refers to a bundle of problems occurring in high dimensional spaces. The most important effect in the sight of clustering is that concepts like “proximity”, “distance”, or “local neighborhood” become less meaningful with increasing dimensionality of a data set (as elaborated e.g. in [31, 69, 9]). As a consequence of these findings, the discrimination between the nearest and the farthest neighbor becomes rather poor with increasing data dimensionality. This is by far a more fundamental problem than the mere performance degradation of algorithms on high dimensional data: The higher the dimensionality of a data set is, the more outliers will be placed inevitably in the set of neighboring objects.

As we will see in this chapter, PCA is very sensitive to outliers. In other words, if the local neighborhood of cluster members or cluster centers to which PCA is applied in order to find the correct subspace of the corresponding cluster contains noise points that do not belong to the cluster, the subspace determination process will be misled. Thus, in view of the “curse of dimensionality”, to successfully employ PCA in correlation clustering in high dimensional data spaces may therefore require more sophisticated techniques of selecting a representative set of neighbors.

This chapter is organized as follows: Section 12.1 presents an analysis of the problems occurring when PCA is used as a basic principle for correlation analysis. We then propose a general framework ready to integrate any of the PCA-based correlation clustering algorithms (see Section 12.2). The application of the framework is exemplarily discussed for ERiC and ORCLUS in Section 12.3. Finally, the impact of the new concepts on the performance of ORCLUS and ERiC is evaluated in Section 12.4.

The material presented in this chapter will appear in [90].

12.1 Problem Analysis

To the best of our knowledge, all correlation clustering algorithms that use PCA as the method to determine the correct subspace of a cluster face the following problem. In order to determine the correct subspace of a cluster, a (considerably large) number of cluster members needs to be identified first such that PCA can be applied to them. On the other hand, in order to identify points of a particular cluster, the subspace of this cluster needs to be determined first. To escape from this vicious circle all algorithms rely on the locality assumption, i.e. it is assumed that the points in the local neighborhood of cluster members or cluster representatives sufficiently reflect the correct subspace of the corresponding cluster such that applying PCA to those neighboring points reports the cluster hyperplane.

As stated above, selecting a meaningful neighborhood becomes more and more difficult with increasing data dimensionality. A neighboring set of points will almost certainly contain outliers, i.e. points that do not belong to the cluster and, thus, are not located on the hyperplane of the cluster. Obviously, these outliers are not helpful to assign a meaningful local correlation dimensionality and orientation. On the other hand, all correlation clustering approaches available rely on an arbitrarily chosen set of neighboring points. We therefore argue to choose a neighboring set of points in a more sophisticated way to enhance the robustness of local correlation analysis and, consequently, to enhance the robustness of correlation clustering algorithms.

12.1.1 Impact of Outliers on PCA

Correlation analysis using PCA is a *least squares fitting* of a linear function to the data. By minimizing the *mean square error*, outliers are emphasized in a way that is not always beneficial, as can bee seen in Figure 12.1. This data set consists of 5 points in a 2D space that are strictly positively correlated and, thus, are located on a common 1D hyperplane plus one additional outlier that is not located on that 1D hyperplane. When applying PCA on these six points and computing the strongest eigenvector of the correspond-

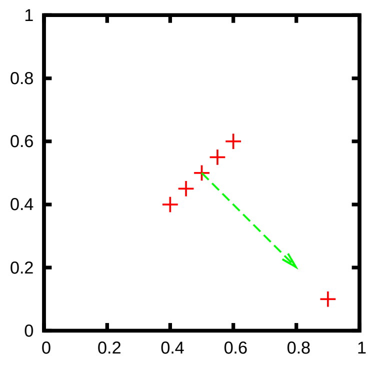


Figure 12.1: Simple data set with 6 points and largest eigenvector after PCA.

ing covariance matrix, the resulting vector is directed towards the outlier (cf. Figure 12.1). This implies that in certain situations, adding only one single extra point to the correlation computation can cause the resulting strongest eigenvector(s) to flip into a completely different direction. Let us note that if the outlier point would have been closer to the other points it would, at a certain distance, not have made any difference on the vector orientation, but this distance threshold for the flip is rather small.

As a consequence, one needs to carefully select the points that are included into the computation of the cluster hyperplane. In addition, one can consider using a modified correlation analysis procedure which is less sensitive to the effect of outliers. In fact, there are obviously multiple strategies to handle these issues. The most obvious one – using outlier detection to remove outliers from the computation – can usually not be applied to this problem because we face the same vicious circle when searching for outliers as we face when detecting cluster points: in order to identify outliers that do not belong to any clusters, the subspaces of the clusters need to be determined first; in order to determine the correct subspace of a cluster, a (considerably large) number of cluster members needs to be identified first such that PCA can be applied to them; etc. Instead, we introduce two ideas to stabilize PCA for correlation clustering. First, we explore a local optimization strategy that handles the problem of picking appropriate neighboring points in a way that

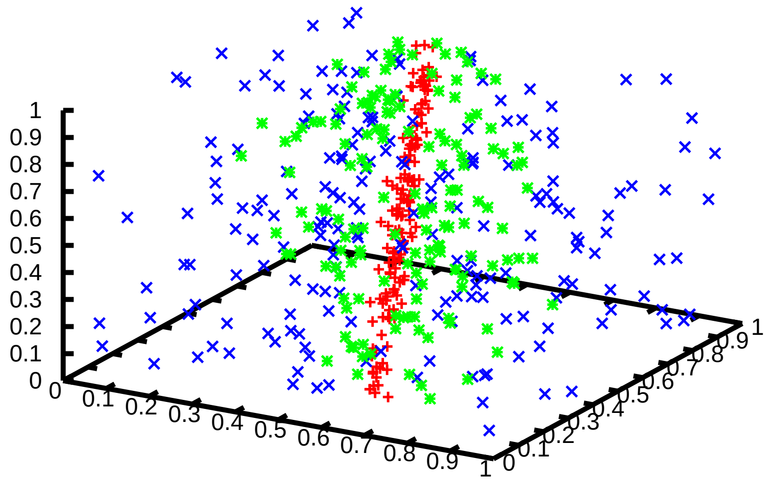


Figure 12.2: Data set with a 2D plane and an embedded 1D line.

is easy to integrate in many correlation clustering algorithms. Second we will add a modified correlation analysis to further stabilize results which is based on the integration of a suitable weighting function into PCA.

12.1.2 Statistic Observations on Data Correlation

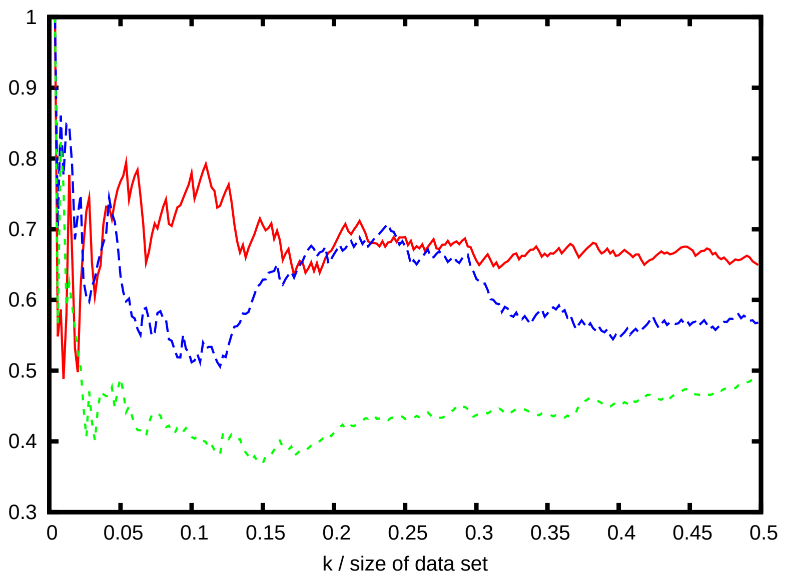
Without loss of generality, we assume that the points on which PCA is applied to find the correct subspace of a particular cluster are selected as the k-nearest neighbors (kNN) of cluster members or cluster representatives. Later, we will discuss the extension of our ideas to methods like ORCLUS that use neighborhood concepts other than kNN.

When comparing the relative strength of the normalized eigenvalues (i.e. the part of the total variance explained by them) computed for the kNN of a particular point w.r.t. increasing values of k (ranging from 0 to 50% of the data set), a behavior similar to that shown in Figure 12.3 can usually be observed. We used a 3D data set shown in Figure 12.2, with a set of 200 outlier points (noise), a correlation cluster of 150 points sharing a common 2D hyperplane (plane), and a correlation cluster of 150 points that are located on a common 1D hyperplane (line) that is embedded into the hyperplane of the 2D cluster. In Figure 12.3 there are three plots in this graph representing the behavior of the eigenvalues of a sample noise point, of a sample point on a 2D, and of a sample point on a 1D line in the data set (embedded within the 2D plane), respectively.

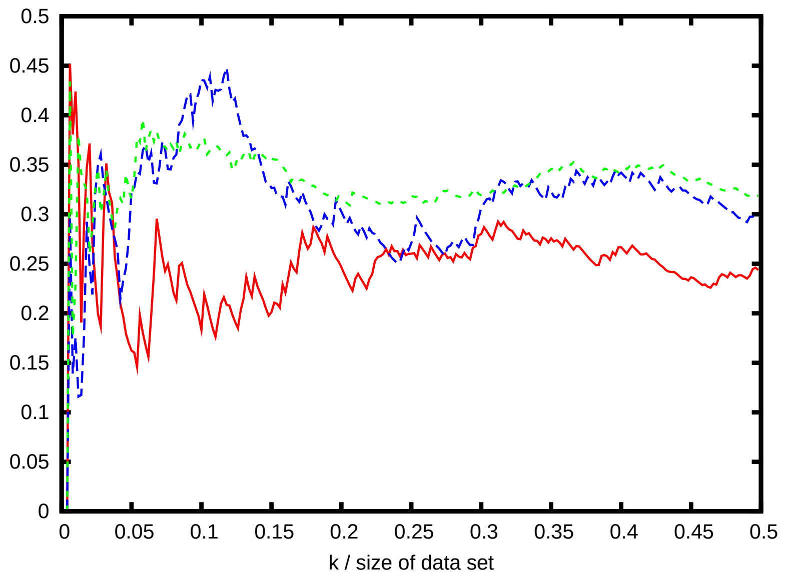
Examining the noise point (green dotted lines in Figure 12.3) we observe a minimum relative strength of the first eigenvalue of about 0.4 for $k = 10\% - 15\%$ (cf. Figure 12.3(a)). Since the minimum possible value for the strongest eigenvector in a 3D data set is $1/3 = 0.33$, the noise point shows approximately no correlation when looking at its kNN with $k = 10\% - 15\%$ of the data set. The second eigenvector (cf. Figure 12.3(b)) shows similar behavior in that particular range of k confirming our conclusions.

For the point in the 1D cluster (red solid lines in Figure 12.3), the first eigenvector (cf. Figure 12.3(a)) explains 80% of the complete variance at around $k = 7\%$, i.e. using this value for k, the kNN of the particular point form the 1D line of the cluster. It is worth noting that the amount of variance explained for the 1D cluster case drops quickly when increasing k beyond this point. The reason for this is that – since the line is embedded in a plane – with increasing k more and more points of the kNN are points from the 2D cluster. As a consequence, the variance explained by the first eigenvector decreases, whereas the variance explained by the second eigenvector increases simultaneously (cf. Figure 12.3(b)). Then, at $k \approx 10\%$, we have again a very high strength of the first eigenvector (less points from the 2D cluster and more points from the 1D cluster are considered), etc. In other words, depending on the value of k, the kNN of the point form the 1D cluster line or the 2D cluster plane.

For evaluating the 2D cluster, the relevant graph (depicting the behavior of the second eigenvector) is shown in Figure 12.3(b). In a 3D data set, a value of around 1/3 would be typical for uncorrelated data and is observable on noise points. For the sample point from the 2D cluster it peaks at almost 45% for about $k = 10\%$. Together with the first graph, this means that the first two eigenvectors explain almost the complete variance at that particular value for k. In other words, for $k = 10\%$, the kNN of this point reflect the 2D



(a) First eigenvector



(b) Second eigenvector

Figure 12.3: Relative strength of eigenvectors.

plane of the cluster sufficiently. Compared to this observation, the variance of the sample point from the 1D cluster embedded in the 2D cluster (red dotted line) along the first two eigenvectors is significantly below the expected value (which is not surprising, having seen that the first eigenvector reaches 80%).

These simple examples illustrate that it is essential to select a sufficient set of points by choosing a suitable value for k. A slight change in k can already make a large difference. Moreover, we have seen that it is rather meaningful to choose even significantly different values of k for different points.

12.2 A General Framework for Robust Correlation Analysis

The above presented considerations induce two important aspects. First, since PCA is a least square fitting and we cannot assume that there are no outliers in the kNN of a point, adjusting the weighting of the points during PCA should improve the results. Second, the selection of points to which PCA is applied can be improved by both micro-adjusting the value of k (to avoid sudden drops in the explained variance) as well as choosing significantly different k for different points in the data set. In the following, we will discuss both aspects in more detail. In fact, our framework for making PCA-based correlation analysis more robust uses both ideas.

12.2.1 Increasing the Robustness of PCA Using Weighted Covariance

As mentioned above, PCA is a common approach to handling correlated data. It is also commonly used for dimensionality reduction by projecting onto the λ strongest (i.e. highest) components. In correlation clustering, PCA is a key method to finding correlated attributes in data.

PCA operates in two steps. In the first step, for any two attributes, i.e. dimensions, d_1 and d_2 the covariance $\mathrm{Cov}(X_{d_1}, X_{d_2})$ of these two dimensions is computed. In the second step, the eigenvectors and eigenvalues of the resulting matrix (which by construction is positive, symmetric and semidefinite) are computed. The computation of eigenvectors and eigenvalues on a symmetric matrix is a standardized procedure which cannot be altered to make the overall process more robust. Instead, the stabilization has to be implemented during the first step.

Given an attribute X, we can model the values of k points in that particular attribute, denoted by x_i for the i-th point, as a random variable. Then, the covariance between two attributes X and Y is mathematically defined as

$$\operatorname{Cov}(X,Y) := E((X - E(X)) \cdot (Y - E(Y))), \tag{12.1}$$

where E is the expectation operator. Usually, one uses the mean of all values of the corresponding attribute as expectation operator, i.e.

$$E(X) = \frac{1}{k}\sum_{i=1}^{k} x_i =: \hat{x}, \tag{12.2}$$

so we have

$$\operatorname{Cov}(X,Y) := \frac{1}{k}\sum_{i=1}^{k}(x_i - \hat{x})(y_i - \hat{y}). \tag{12.3}$$

Obviously, all data points are treated equally in this computation. But given that we want to reduce the effect of outliers, it is more appropriate to use a different expectation operator. Given arbitrary weights ω_i for all points i ($1 \leq i \leq k$ and $\Omega := \sum_{i=1}^{k} \omega_i$), we can define a new expectation operator

$$E_\omega(X) := \frac{1}{\Omega}\sum_{i=1}^{k} \omega_i x_i =: \hat{x_\omega}. \tag{12.4}$$

With this new expectation operator, we can give each point in kNN a different weight. In particular, we can give potential outliers a smaller weight. Using $E_\omega(X)$, we can compute the covariance as given below.

$$\operatorname{Cov}_\omega(X,Y) := \frac{1}{\Omega}\sum_{i=1}^{n} \omega_i (x_i - \hat{x}_\omega)(y_i - \hat{y}_\omega). \tag{12.5}$$

Steiner's translation still applies, which leads to the following slightly simpler equation.

$$\operatorname{Cov}_\omega(X,Y) = (\frac{1}{\Omega}\sum_{i=1}^{n} \omega_i x_i y_i) - (\frac{1}{\Omega}\sum_{i=1}^{n} \omega_i x_i) \cdot (\frac{1}{\Omega}\sum_{i=1}^{n} \omega_i y_i). \tag{12.6}$$

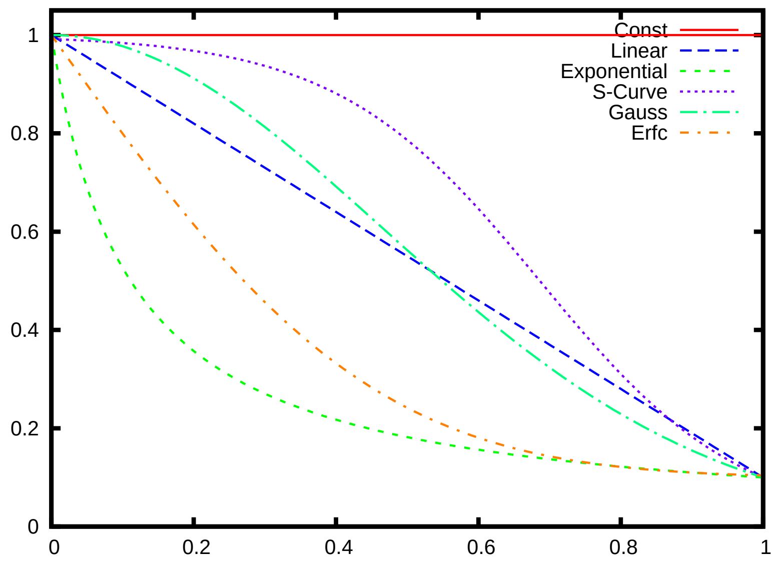


Figure 12.4: Some weight functions.

This form is particularly nice for computation. It is also trivial to prove that if $\omega_i = 1$ for all i, we have $\mathrm{Cov}(X, Y) = \mathrm{Cov}_\omega(X, Y)$. If a point i is assigned the weight $\omega_i = 2$, the result would be the same as if we had two points with the same coordinates as i. If a point i is weighted by $\omega_i = 0$, the result is the same as if point i had not been included in the computation at all.

We can now use arbitrary weighting functions to calculate the weights to be used. Obviously, we again have the dilemma that we do not know which points are outliers and need to get assigned a lower weight. However, since all algorithms use the locality assumption, we can make the following considerations: On the one hand, it is usually very likely that taking the local neighborhood of points includes a lot of outliers. But on the other hand, the neighbors that are near to the query point will more likely be cluster members than the neighbors that are farther apart from the query point. So a distance-based weighting function will most likely weight cluster points higher and outliers lower.

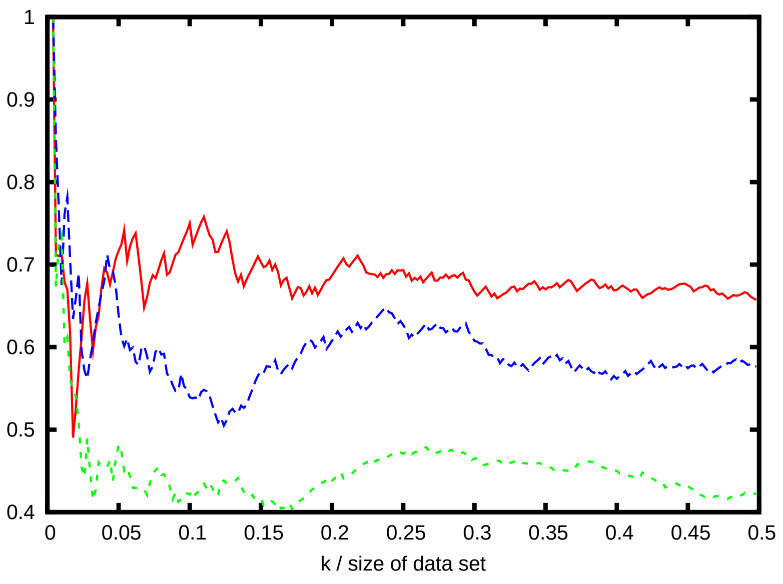
Some examples of distance-based weighting functions are given in Figure 12.4. We have chosen parameters such that the value at $x = 0.0$ is about

$f(0.0) \sim 1.0$ and at $x = 1.0$ it is about $f(1.0) \sim 0.1$. Weights too close to 0.0 are not very useful, because then, these points are not considered for the computation at all. The example weighting functions we have used in our experiments (cf. Figure 12.4) include a constant weighting of 1.0 (solid red line in Figure 12.4), a linearly decreasing function ranging from 1.0 to 0.1 (dashed blue line in Figure 12.4), an exponential fall-off (green dashed line in Figure 12.4), a sigmoid-curved fall-off (violet dotted line in Figure 12.4), a Gauss function (green dashed-dotted line in Figure 12.4), and the complementary Gauss Error Function *Erfc* (red dashed-dotted line in Figure 12.4). The last one is a function well-known from statistics related to normal distributions and, thus, probably the most sound choice.

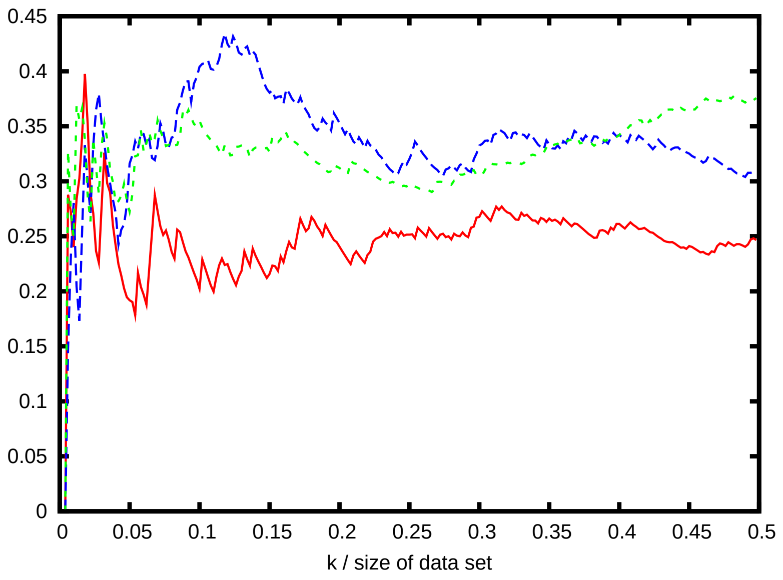
In our experiments, all of the alternative weighting functions (except the constant weight) lead to similar improvements so there is no reliable measure or significance to establish a ranking between the different weighting functions. In fact, it is plausible that different functions are appropriate for different underlying causes in the data or assumptions in the clustering process (e.g. clustering algorithms assuming a Gauss distribution might benefit best from a Gaussian weighting function).

For distance-based weighting functions, several tasks arise. We have chosen to scale distances such that the outermost point has a distance of 1.0, i.e. a weight of 0.1, ensuring that this point has still some guaranteed influence on the result. This choice is somehow arbitrary, but it has at least the benefit of fairness. On the other hand, this fairness comes at the cost that all weights depend on the outermost point. When points are selected using a range query, the query range could offer a better normalization. When an incremental computation is desired, a completely different choice might be appropriate. Additionally, we are computing weights based on the distance to a query point. This is appropriate for situations where the data are obtained via kNN or ε-neighborhoods. When computing the correlation for an arbitrary set of points, the distance might need to be computed from the centroid or medoid of that set.

In the above described toy example of five cluster points plus one outlier



(a) First eigenvector.



(b) Second eigenvector.

Figure 12.5: Relative strength of eigenvectors (with *Erfc* weight).

(cf. Figure 12.1), the observed sensitivity to that outlier is significantly decreased, given that the outlier will only be weighted at around 0.1. Applying the weighting function to the 3D example data set of Figure 12.2 we also observe an increased robustness of the correlation analysis. Figures 12.5(a) and 12.5(b) depict the effect of a weighted covariance on the relative strength of the first eigenvector and the normalized sum of the first two eigenvectors, respectively, using the *Erfc* weighting. Compared to Figures 12.3(a) and 12.3(b) we can derive that many of the sudden drops have been erased, while the overall shape is well preserved. Especially for higher values of k, sudden jumps have mostly disappeared. Therefore, this measure is useful to avoid choosing a particularly bad value of k, i.e. a k where the kNN of the particular point do not reflect the correct subspace of the corresponding cluster, by somewhat averaging with neighbors. Peaks usually are shifted towards a slightly higher value of k. This is natural since the added points are weighted low at first.

12.2.2 Auto-tuning the Local Context of Correlation Analysis

Graphs such as Figure 12.3(a) show that even small differences in k can lead to significantly different results. Therefore, it is reasonable not to use a fixed value of k, i.e. a fixed number of neighboring points, but rather to adjust the value of k for each point separately. For example, one can use a globally fixed number of neighbors k_{max} and then individually select for each point the $k \leq k_{max}$ neighbors that are relevant for the particular point. As far as k_{max} is sufficiently large, we should in general be able to select a reasonable k, so that this strategy produces accurate results. Of course there are different strategies of selecting k. Since there are $O(2^{k_{max}})$ subsets of the given k_{max} points that could be used, simply trying all combinations of subsets of k points ($1 \leq k \leq k_{max}$) is not feasible. Probably the easiest strategy of $O(k_{max})$ complexity is to test for any k ($1 \leq k \leq k_{max}$) only the k nearest points, resulting in k_{max} tests. The next question that arises is how to evaluate the results of the k_{max} tests in order to report the best

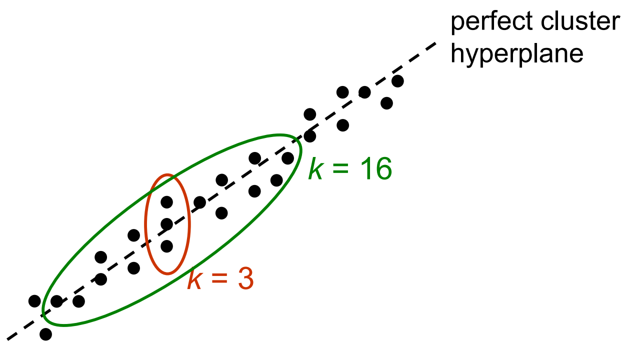


Figure 12.6: Problems with jitter.

value for k. The obvious strategy of returning the result that maximizes the relative strength of eigenvalues has shown to be not very reliable because of jitter: one particular k value could result in a "perfect" hyperplane consisting mainly of points that form a subspace completely different to the subspace of the cluster. Figure 12.6 illustrates this effect: using only the three points in the red ellipsoid, we will hardly find the correct hyperplane of the cluster although all those three points are cluster members because they do not fit the subspace perfectly. Rather, the three points perfectly form a different line so the relative strength of the first eigenvalue will be very high (appr. 100%). In fact, we are more interested in a range of k values where we have a high and stable relative strength of eigenvalues, so we need a more elaborate filtering.

In our evaluations, we have chosen the strategy to use the k nearest points for correlation analysis, with $k_{min} \leq k \leq k_{max}$, where k_{min} is a minimum number of points such that the PCA is at least somewhat sensible at this data set dimension. The motivation behind the introduction of the lower bound k_{min} is that we need at least λ points to span a λ-dimensional hyperplane and $3 \cdot \lambda$ has been considered as a lower bound of points such that the detection of a λ-dimensional hyperplane by PCA is trustworthy rather than arbitrary. To avoid jitter and outlier effects, we use a sliding window to apply a dimensionality filter and average the variance explained by the largest eigenvalues.

Let

$$\mathrm{ex}(E,\lambda) := \frac{\sum_{i=1}^{\lambda} e_i}{\sum_{i=1}^{d} e_i} \tag{12.7}$$

be the relative amount of variance explained by λ eigenvalues $E = \{e_i\}$ representing a hyperplane of dimensionality λ. Most correlation clustering algorithms rely on a level of significance $\alpha \leq 1$ to decide how many eigenvectors explain a significant variance and, thus, span the hyperplane of the cluster. Intuitively, the eigenvectors are chosen such that the corresponding eigenvalues explain more than α of the total variance. The number of those eigenvectors is called *local dimensionality* (of a cluster), denoted by λ_E, formally

$$\lambda_E = \min_{\lambda \in \{1 \ldots d\}} \{\lambda \mid \mathrm{ex}(E,\lambda) \geq \alpha\} . \tag{12.8}$$

Let us note that almost all correlation clustering algorithms use this notion of local dimensionality. Typical values for α are 0.85, i.e. the eigenvectors that span the hyperplane explain 85% of the total variance along all eigenvectors.

As indicated above, for filtering out the best value of k, we are intuitively interested in a value where (i) the local dimensionality λ is stable, i.e. increasing or decreasing k by a small degree does not affect the value of λ, and (ii) $\mathrm{ex}(E,\lambda)$ is maximal and stable, i.e. increasing or decreasing k by a small degree does not affect the value of $\mathrm{ex}(E,\lambda)$. The motivation behind these considerations is that the value of k that fulfills both properties leads to the determination of a robust hyperplane, that maximizes the variance along its axis. In other words, using the neighbors determined by k, the hyperplane reflects all of these neighbors in a best possible way and there are most likely only very few neighbors that are outliers to this hyperplane. In addition, increasing or decreasing k, i.e. adding or deleting few neighbors, does not affect the correlation analysis.

To find the value of k that meets both properties, we determine $\mathrm{ex}(E,\lambda)$ for all $k_{min} \leq k \leq k_{max}$. We then use a sliding window $W = [k_l, k_u]$ and choose $k = (k_l + k_u)/2$ such that for all k' in W (i.e. $k_l \leq k' \leq k_u$) the local dimensionality λ is the same and the average of $\mathrm{ex}(E_{k'},\lambda)$ is maximized. Additionally, if this maximum is at the very beginning or end of our search range (i.e. $k_l = k_{min}$ or $k_u = k_{max}$), we discard it. We can still obtain multiple maxima, one for each dimensionality λ. In this case we pick the lowest like all correlation clustering algorithms aiming at finding the lowest dimensional subspace clusters. Those are the most interesting ones since they involve the largest set of correlations among attributes.

12.3 Application to Existing Approaches

In the following, we discuss how our concepts can be integrated into existing correlation clustering algorithms in order to enhance the quality of their results. Exemplarily, we show this integration with two different types of algorithms, the latest density-based algorithm ERiC and the k-means-based algorithm ORCLUS.

12.3.1 Application to Density-based Correlation Clustering Algorithms

The integration of our concepts into ERiC is rather straightforward. ERiC determines for each data point p the subspace of the cluster to which p should be assigned (hereafter called the subspace of p). The subspace of p is computed by applying PCA to the kNN of p where k needs to be specified by the user.

Using our concepts, we can simply replace the parameter k by the global maximum k_{max} of neighbors that should be considered. Both the weighting and the auto-tuning can then be applied directly when computing the subspace of p. First, from the k_{max}NN of p, the optimal $k_p \leq k_{max}$ for detecting the subspace of p is determined as described in Section 12.2.2 based on a weighted covariance as described in Section 12.2.1. Second, the subspace of p is computed by applying PCA using a weighted covariance on the k_pNN of p (cf. Section 12.2.1).

The integration of our concepts into other density-based algorithms like

COPAC, HiCO, and 4C can be done analogously.

12.3.2 Application to Partitioning Correlation Clustering Algorithms

ORCLUS determines the subspace of each cluster C by applying PCA to the local neighborhood of the center of C, denoted by r_C. The local neighborhood of r_C includes the set S_C of all points that have r_C as their nearest cluster representative.

Using our concepts, we can simply consider S_C as the maximum set of points that should be considered for PCA, i.e. $k_{max} = |S_C|$. Both the weighting and the auto-tuning can then be applied directly when computing the subspace of C. First, from the S_C, the optimal $k_C \leq k_{max}$ for detecting the subspace of C is determined as described in Section 12.2.2 based on a weighted covariance as described in Section 12.2.1. Second, the subspace of C is computed by applying PCA using a weighted covariance on the k_C points in S_C that are closest to r_C (cf. Section 12.2.1).

12.4 Evaluation

12.4.1 Evaluation Methodology

In order to evaluate the results of our novel concepts integrated into ERiC and ORCLUS, we generated artificial data sets with a well-defined gold standard, i.e. we defined certain data distributions and all points in our data set are assigned to the distribution with the maximum density in that particular point. Since both ERiC and ORCLUS have different properties and, here, we are not interested in judging which algorithm is better for which data set, we generated different data sets for each algorithm.

To evaluate the quality of the clustering, we employ a pair-counting F-measure, considering the noise points to be a cluster on its own. This means that any two points in the data set form a pair if they belong to the same cluster (or noise). Let $C = \{C_i\}$ be a clustering (with C_i being the clusters in C, including the noise cluster). Then $P_C := \{(a, b) \mid \exists C_i : a \in C_i \wedge b \in C_i\}$ is the set of pairs in clustering C. The F-measure to evaluate how good a clustering C matches the gold standard D is then defined as

$$F(C, D) := \frac{2 \cdot |P_C \cap P_D|}{2 \cdot |P_C \cap P_D| + |P_C \setminus P_D| + |P_D \setminus P_C|}.$$

Obviously, $F(C, D) \in [0, 1]$, where $F(C, D) = 1.0$ means that the clustering C is identical to the gold standard D.

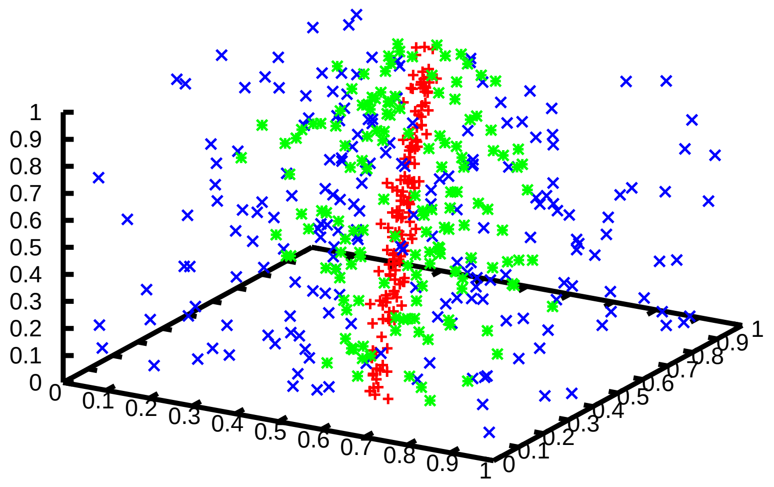
12.4.2 Synthetic Data

For evaluating the influence of our novel methods on both ORCLUS and ERiC, we used several synthetic data sets ranging from 3 to 100 dimensions. In the following discussion, we focus on some lower dimensional data sets for a clear presentation.

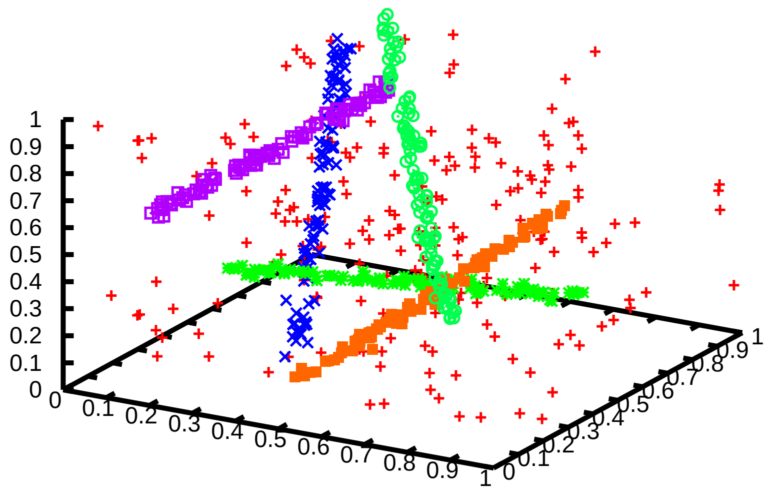
ERiC

We first focus on two 3D synthetic data sets that can be seen in Figure 12.7. Figure 12.8(a) gives the results for data set DS1 shown in Figure 12.7(a). We plotted the F-measure of the compared algorithms along the y-axis and varied the parameters k and k_{max} along the x-axis. The blue line represents the results of the unmodified ERiC algorithm. Obviously the choice of k is nontrivial, a value of about $k = 34$ gives the best results. The violet dotted line is the result when using the *Erfc* weight in PCA. Obviously, the results are significantly better, and any k in $35 < k < 65$ gives good results. Therefore choosing a good k has become a lot easier using only the weighting approach. The green line with the short dash-dot pattern depicts the result of ERiC using a Gauss weight. As it can be seen, the results using a simple Gaussian weighting do not significantly differ from the *Erfc* weighting results.

The remaining three lines show the results of ERiC when using the auto-tuning of the parameter k, i.e. for each point the optimal $k \leq k_{max}$ is de-

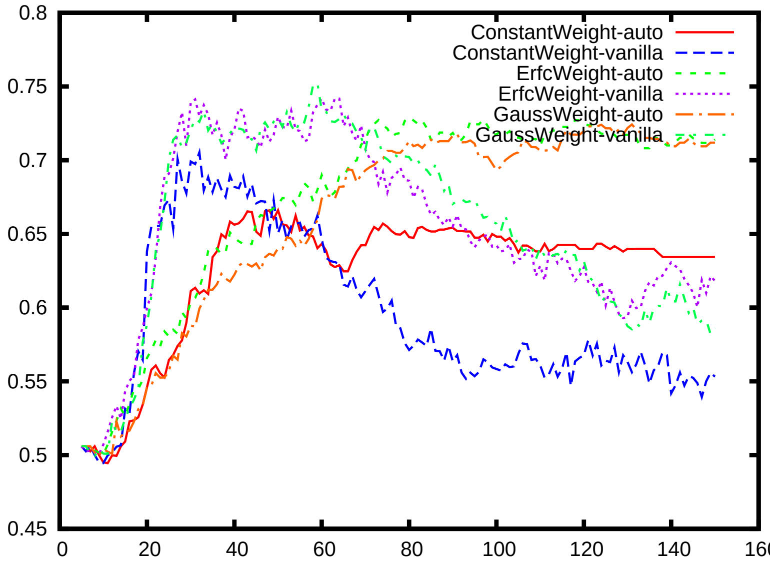


(a) DS1: a 1D lines (150 points) embedded within a 2D plane (150 points) plus 200 points noise.

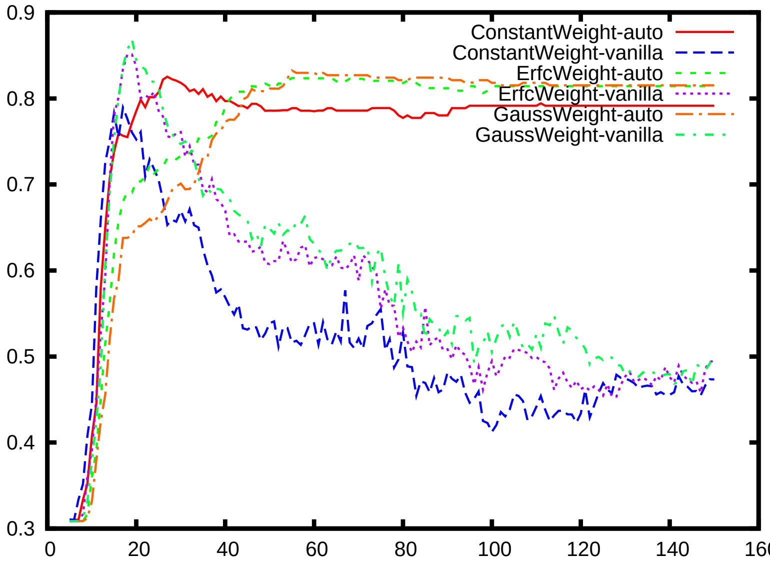


(b) DS2: five 1D lines (100 points each) plus 200 points noise.

Figure 12.7: 3D synthetic data sets used for evaluating ERiC.



(a) DS1.



(b) DS2.

Figure 12.8: Results of ERiC with different weight functions and auto-tuning on 3D synthetic data sets.

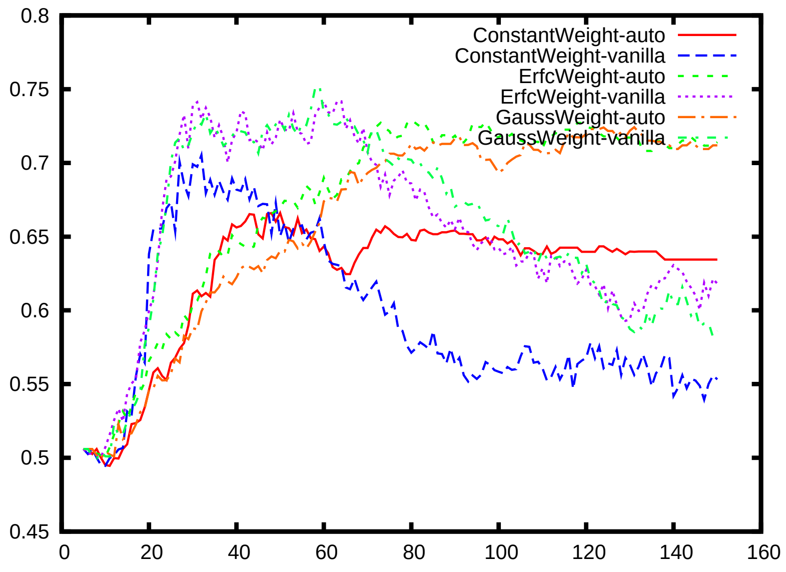


Figure 12.9: Results of ERiC with different weight functions and auto-tuning on a sample 10D synthetic data set.

termined separately (for these graphs, the x-axis represents the chosen k_{max} value). The red line is using the traditional PCA without any weighting, while the dashed green and the orange line with the long dash-dot pattern represent the results using the *Erfc* and Gauss weights, respectively. The results show that k_{max} simply needs to be chosen high enough in order to achieve reasonably good results. While these results do not reach the results of choosing the optimum k (which is not possible without knowing the gold standard), they approach the optimal value quite well. This observation dramatically simplifies the choice of the k/k_{max} parameter.

Figure 12.8(b) depicts the results on DS2 shown in Figure 12.7(b). Overall, the results on DS1 and DS2 are comparable. However, given that every point has just one "sensible" dimensionality – the other data set had points that had both a sensible 1D and 2D context – and the noise level is not as high, the effect of the weighted PCA on DS2 is not as high as on DS1. Since increasing the noise level will increase the difference between the non-weighted and weighted graphs, the weighting is especially interesting for noisy (e.g. higher dimensional) data.

All observations that could be made for the two 3D data sets could also be made for higher dimensional data sets. For example, Figure 12.9 shows the results of ERiC with different extensions for a sample 10D data set. Again, the version of ERiC using an *Erfc* weighted PCA in combination with the auto-tuned selection of k achieved the best overall F-measure. Also, as long as k_{max} is chosen sufficiently high, we get rather accurate results.

In summary, we observed that in all cases, the combination of the *Erfc* weighted PCA and the auto-tuned selection of k considerably increased the F-measure of the resulting clustering and significantly reduced the complexity of selecting sufficient input parameters compared to the original ERiC algorithm.

ORCLUS

The results of ORCLUS are harder to evaluate, because the results of ORCLUS depend on the order in which the data points are processed. Therefore, we generated 100 permutations of the original data, applied ORCLUS with optimal parameters to all of them, and averaged the results. The data set used in these computations was a 10-dimensional data set, containing 10 clusters of dimensionalities 2 to 5. The results are given in Table 12.1.

Each of these values was obtained by running ORCLUS and its variants on the same 100 permutations of the input data set and averaging the resulting F-measure values. The standard deviation over the 100 resulting F-measure values is given to show the dependence of ORCLUS on picking good seeds. It can be observed that the benefits of using a weighted PCA are smaller (≈ 0.02) than those of using an auto-tuning PCA (≈ 0.09) and the combination of both actions further improves the results. Interestingly, in this experiment, a linear weighting function is slightly better (by up to 0.02) than a Gaussian or *Erfc* weighting. However, in general on different data sets, there is no significant difference observable comparing different weighting functions. In summary, using our novel concepts, the F-measure on this data set is improved by approximately 0.1 corresponding to a 10% quality boost.

Table 12.1: Impact of the integration of our novel concepts into ORCLUS.

Variant	**Avg. F-measure**	**St. Dev.**
ORCLUS	0.667	0.046
ORCLUS + Gauss weight	0.684	0.055
ORCLUS + Exponential weight	0.676	0.054
ORCLUS + *Erfc* weight	0.683	0.061
ORCLUS + Linear weight	0.686	0.056
ORCLUS + Auto	0.751	0.070
ORCLUS + Auto + Gauss	0.763	0.069
ORCLUS + Auto + Exponential	0.754	0.075
ORCLUS + Auto + *Erfc*	0.754	0.075
ORCLUS + Auto + Linear	0.771	0.078

Table 12.2: Results on NBA data using ERiC with autotuning and *Erfc* weighting.

cluster ID	dim	Description
1	4	“go-to-guys”
2	4	guards
3	4	reserves
4	5	small forwards

12.4.3 Real-world Data

We applied the enhanced version of ERiC (using autotuning and *Erfc* weighting) on a data set containing average career statistics of current and former NBA players[1]. The data contains 15 features such as “games played” (G), “games started” (GS), “minutes played per game” (MPG), “points per game” (PPG), etc. for 413 former and current NBA players. We detected 4 interesting clusters each containing players of similar characteristics (cf. Table 12.2). In addition, several players were assigned to the noise set. Cluster 1 contains active and former superstars like Michael Jordan, Scottie Pippen,

[1]obtained from `http://www.nba.com`

Table 12.3: Clustering results on Metabolome data using ERiC with auto-tuning and *Erfc* weighting.

cluster ID	dim	Description
1	10	PKU
2	10	controll
3	11	PKU
4	12	PKU
5	13	PKU

Gary Payton, Allen Iverson, Larry Bird, Dominique Wilkins, and LeBron James, etc. The second cluster features point and shooting guards, e.g. Reggie Miller, Nate McMillan, Tim Hardaway, John Stockton, Steve Kerr, Glen Rice, Steve Nash, Derek Anderson, and Eddie Jones, among others. A third cluster contains only very few players that are not so well-known because they are usually reserves. The fourth cluster consists of small forwards such as Bryon Russell, Detlef Schrempf, Morris Peterson, Sean Elliott, Josh Howard, Tom Chambers, Jerome Kersey, and Al Harrington. Let us note that we also applied the original ERiC algorithm (without the extensions) to the NBA data set but could not get any clear clusters. In summary, using our novel concepts, the algorithm ERiC is now able to detect some meaningful clusters on the NBA set.

In addition, we applied our novel concepts in combination with ERiC to the Metabolome data set of [92] consisting of the concentrations of 43 metabolites in 20,391 human newborns. The newborns were labeled according to some specific metabolic diseases. The data contain 19,730 healthy newborns ("control"), 306 newborns suffering from phenylketonuria ("PKU"), and 355 newborns suffering from any other diseases ("other"). The results are depicted in Table 12.3. As it can be seen, we could separate several of the newborns suffering from PKU from the other newborns. Again, the original version of ERiC could not find any comparatively good results.

Part IV

Global Correlation Clustering

In the previous part, several contributions to the field of correlation clustering have been discussed. All original approaches to correlation clustering proposed in Part III are based on a marriage of the density-based cluster paradigm and PCA. Clustering following the density-based paradigm is either flat (as it applies to the algorithms in Chapters 8 and 9) or hierarchical (as the algorithms discussed in Chapters 10 and 11). The hierarchy of correlation clusters, however, is not only a containment hierarchy of clusters but also of the corresponding subspaces.

While Chapter 12 tackled some weak points common to all approaches to correlation clustering based on PCA, there remain intrinsic drawbacks in the application of these PCA-and-density-based approaches to high dimensional data. These have been basically discussed in Part II and can be summarized in the so called "locality assumption": The application of the density-based paradigm to clustering in arbitrarily oriented subspaces of high dimensional data assumes the subspace preference of a point being manifest in its local neighborhood. This locality assumption is presumably the main reason why all these approaches discussed in the previous part are suitable to cluster data sets of moderate dimensionality only. The same applies to the algorithm ORCLUS [11] which is not density-based but partitioning (a k-medoid-like approach). Nevertheless, PCA is applied on a local selection of points (the Voronoi-parcel of the respective cluster-medoid) in order to ascertain the suitable subspace.

This part now presents a first attempt to overcome this limitation in principle and leads therefore to a global approach to correlation clustering.

We will first recall the difficulties in naïvely accepting the "locality assumption" (Chapter 13). Afterwards we present a fundamentally new approach for a global correlation analysis (Chapter 14).

Chapter 13

Local versus Global Correlation Clustering

13.1 Motivation

Subspace clustering is a data mining task which has attracted considerable attention during the last years. There are two main reasons for this popularity. Firstly, conventional (full space) clustering algorithms often fail to find useful clusters when applied to data sets of higher dimensionality, because typically many of the attributes are noisy, some attributes may exhibit correlations among another, and only few of the attributes really contribute to the cluster structure. Secondly, the knowledge gained from a subspace clustering algorithm is much richer than that of a conventional clustering algorithm. It can be used for interpretation, data compression, similarity search, etc.

As discussed in Part II, we can distinguish between subspace clustering algorithms for axis-parallel subspaces [13, 80, 10, 114, 34, 2, 3] and those for subspaces which are arbitrarily oriented (called *oriented clustering*, *generalized subspace clustering*, or *correlation clustering*, e.g., [11, 35]). In both cases, the data objects which are grouped into a common subspace cluster, are very dense (i.e., the variance is small) when projected onto the hyperplane which is perpendicular to the subspace of the cluster (called the *perpendicular* *space plane*). The objects may form a completely arbitrary shape with a high variance when projected onto the hyperplane of the subspace in which the cluster resides (called the *cluster subspace plane*). This means, that the objects of the subspace cluster are all *close* to the cluster subspace plane. The knowledge, that all data objects of a cluster are close to the cluster subspace plane is valuable for many applications: If the plane is axis-parallel, this means that the values of some of the attributes are, more or less, constant for all cluster members. The whole group is characterized by this constant attribute value, an information which can definitely be important for the interpretation of the cluster. This property may also be used to perform a dedicated dimensionality reduction for the objects of the cluster and may be useful for data compression (because only the higher-variance attributes must be stored at high precision individually for each cluster member) and similarity search (because only the high-variance attributes need to be individually considered for the search and an index needs only be constructed for the high-variance attributes).

If the cluster subspace plane is arbitrarily oriented, the knowledge is even more valuable. In this case, we know that the attributes which define the cluster subspace plane, have a complex dependency among each other. This dependency defines a rule, which again characterizes the cluster and which is potentially useful for cluster interpretation. Similarly to the case of axis-parallel clusters, this dependency rule may also be used for dimensionality reduction, data compression, similarity search, and indexing. Consider, for example, Figure 13.1 which contains two general subspace clusters in a very noisy environment. For each of the subspace clusters, we know that the x and y coordinates are approximately linearly dependent from each other ($y \approx m_i \cdot x + t_i$), and, therefore, only one of them needs to be stored at full precision, indexed, etc. Furthermore, the knowledge of the degree of dependency, as well as the slope and intercept may be important for the interpretation of the cluster in the context of the application.

One well-known effect of the "curse of dimensionality" is the correlation among attributes in high dimensional data. While full dimensional clustering approaches are easily misled by these correlations, generalized subspace

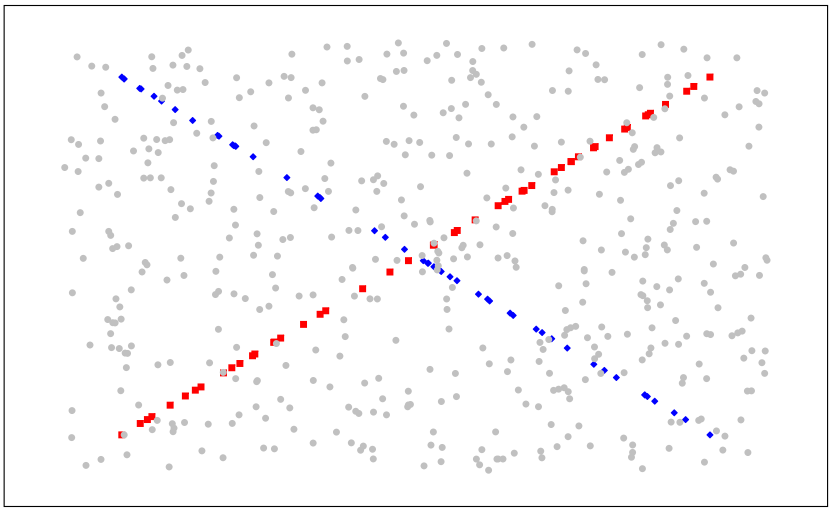


Figure 13.1: Data set with two non-dense general subspace clusters in a noisy environment

clustering approaches, hence also called *correlation clustering*, make use of this effect to identify clusters in subspaces of arbitrary dimensionality. However, finding axis-parallel or generally oriented subspace clusters is not a trivial task. The number of possible axis parallel subspaces is exponential in the number of dimensions, and the number of general subspaces is even infinite. Therefore, a complete enumeration of all possible subspaces to be checked for clusters is not feasible. Consequently, all previous solutions rely on specific assumptions and heuristics, and try to find promising subspaces during the clustering process, for instance in an iterative optimization. We will see that this previous approach of *learning* suitable subspaces works well if (but only if) subspace clusters are locally well separated and no outlier objects (belonging to no cluster) exist. In the presence of outliers in the local neighborhood of cluster points or cluster representatives in the entire feature space, most previous subspace clustering algorithms fail to detect subspace clusters, because the algorithms try to find suitable subspaces for each cluster from the local neighborhood of cluster points or cluster representatives in the entire feature space. This fundamental assumption all existing approaches to correlation clustering are based upon is called the *"locality assumption"*. Outliers in the neighborhoods, that do not belong to the corresponding cluster prevent the algorithms from finding suitable subspaces, and the absence of a precise subspace prevents the algorithm from effectively filtering out the outliers.

In high dimensional spaces, however, where distances cannot be used to differentiate between near and far points, the concept of local neighborhoods is meaningless [31, 69, 9]. Consequently, the neighborhoods of cluster points or cluster representatives will contain a large number of outliers that do not belong to the corresponding cluster. However, those problems arise even if the number of outliers is very small (e.g. 5-10 outliers in the complete data set). Thus, an environment of heavy noise such as that of Figure 13.1 is completely out of the scope of previous subspace clustering methods even in lower dimensional data spaces, as we will discuss more deeply in Section 13.2 for locally optimizing approaches such as ORCLUS [11] and for density-based approaches such as 4C [35] and its variants (see Part III).

13.2 The "Locality Assumption" in Existing Correlation Clustering Algorithms

Existing approaches for subspace clustering rely on certain heuristics that use specific assumptions to shrink down the search space and thus to reduce the runtime complexity. However, if these assumptions are not true for a given data set, the methods will either fail to detect any suitable patterns or exhibit an exponential runtime.

Many subspace clustering algorithms (e.g. [13, 80, 10, 114, 34, 2, 3]) assume that the subspace clusters are axis-parallel. Otherwise, they will not find any pattern. Pattern-based subspace clustering algorithms (e.g. [41, 144, 138, 110, 96]) are limited to find only clusters that represent pair-wise positive correlations in the data set. In contrast, arbitrarily oriented hyperplanes (subspace clusters) may also represent more complex or negative correlations.

Here, we focus on the generalized problem of finding arbitrarily oriented subspace clusters. All existing algorithms for this problem assume that the cluster structure is significantly dense in the local neighborhood of the cluster centers or other points that participate in the cluster. In the context of high dimensional data this "locality assumption" is rather optimistic. Theoretical considerations [69] show that concepts like "local neighborhood" are not meaningful in high dimensional spaces because distances can no longer be used to differentiate between points. This is a consequence of the well-known curse of dimensionality.

ORCLUS [11] is based on k-means and iteratively learns the similarity measure capturing the subspace containing a given cluster from the points assigned to the cluster in each iteration by applying PCA on these points. Since the algorithm starts with the Euclidean distance, the algorithm learns the subspaces from the local neighborhood of the initial cluster centers. However, if this local neighborhood contains some noise or the clustering structure is too sparse within this local neighborhood, the learning heuristic will be misled because PCA is rather sensitive to outliers. In those cases, ORCLUS will fail to detect meaningful patterns. These considerations accordingly apply to the method proposed in [39] which is a slight variant of ORCLUS designed for enhancing multi-dimensional indexing.

4C [35] integrates PCA into density-based clustering. It evaluates the Euclidean neighborhood of each point p to learn the subspace characteristics in which p can be clustered best. Similar to ORCLUS, 4C thus relies on the assumption that the clustering structure is dense in the entire feature space. Otherwise 4C will also fail to produce meaningful results. The same holds true to some variations of 4C like COPAC [6], HiCO [7], and ERiC [5], and also for robustified versions of these algorithms as described in [90]. (See Part III for a detailed description of these algorithms.)

The method CURLER [136] merges the clusters computed by the EM algorithm using the so-called co-sharing level. The resulting clusters need not to represent linear correlations. Rather, any dense pattern in the data space is found that may represent a more complex, not necessarily linear correlation. CURLER also relies on the assumption that the subspace clustering structure is dense in the entire feature space because both the generation as well as the merging of micro-clusters uses local neighborhood information.

In summary, as discussed in Part II, Chapter 7, existing approaches to the correlation clustering problem tackle only one problem out of the bundle of problems known as "curse of dimensionality", namely the occurring correlation of attributes. In view of the remaining problems (esp. *irrelevance of neighborhood* and *irrelevant attributes*), the "locality assumption" cannot lead to satisfactory solutions in high dimensional data. This may be the reason why all these algorithms appear suitable only for data sets of moderate dimensionality.

To overcome the restrictions of the "locality assumption" for correlation clustering, a truly "global" search for correlations among attributes is required. A "global" approach should not be obfuscated by irrelevant attributes or meaningless neighborhood queries. The next chapter is dedicated to describe a first shot in this direction.

Chapter 14

Correlation Clustering Based on the Hough-transform

Obviously, the “locality assumption” that the clustering structure is dense in the entire feature space and that the Euclidean neighborhood of points in the cluster or of cluster centers does not contain noise is a very strict limitation for high dimensional real-world data sets. In [69] the authors show that in high dimensional spaces, the distance to the nearest neighbor and the distance to the farthest neighbor converge. As a consequence, distances can no longer be used to differentiate between points in high dimensional spaces and concepts like the neighborhood of points become meaningless. Usually, although many points share a common hyperplane, they are not close to each other in the original feature space. In those cases, existing approaches will fail to detect meaningful patterns because they cannot learn the correct subspaces of the clusters. In addition, as long as the correct subspaces of the clusters cannot be determined, obviously outliers and noise cannot be removed in a preprocessing step.

In this chapter, we propose to use the ideas of the Hough transform [71] to develop an original principle for characterizing the subspace containing a cluster. This way, correlation clusters are sought in a truly “global” way, thus overcoming the limitations of the “locality assumption”.

This chapter is organized as follows: The fundamental concepts of the Hough-transform are introduced in Section 14.1. Based on these ideas, a generalization to d-dimensional data and the basic principle to make use of these concepts for correlation clustering are sketched in Section 14.2. This principle enables us to transform the task of subspace clustering (in data space) into a grid-based clustering problem (in parameter space). Unlike grid-based methods operating directly in the data space, our method does not suffer from grid resolution and grid positioning problems. In order to perform this transformation, we first need to define the boundaries of the grid (cf. Section 14.3). In this step, some rather technical considerations are required. These are described in detail in Section 14.4 so as to not obstruct the flow of reading. Then we will show how to identify dense grid cells that represent potential subspace clusters (cf. Section 14.5). Since the parameter space is d-dimensional for a d-dimensional data space, finding dense grid cells becomes rather costly for higher dimensional data sets. Thus, we will propose a more efficient search strategy for finding regions of interest in the parameter space (cf. Section 14.6). An important step in the clustering process is a recursive descent in order to find lower dimensional clusters. We describe this descent in more detail in Section 14.7. We will also discuss how this recursive descent can be used to derive a hierarchy of subspace clusters (cf. Section 14.8). We will also summarize our subspace clustering algorithm CASH (Clustering in Arbitrary Subspaces based on the Hough transform) and discuss some of its properties (cf. Section 14.9). Finally, an experimental evaluation is presented in Section 14.10.

The concepts presented in this chapter are partially published in [1].

14.1 The Hough-transform

The basic Hough transform has been introduced in the computer graphics community to address the problem of finding linear segments in pictures (especially straight lines) by [116]. Most work focuses on discretized 2D data. The key idea is to map each point of a 2D picture (or data space $\mathcal{D}$) such as

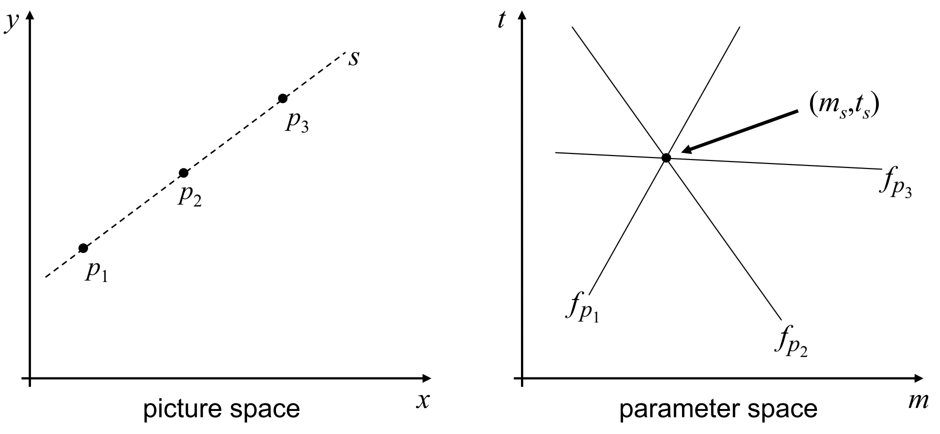


Figure 14.1: Hough transform from picture space to parameter space using slope and intercept parameters.

a pixel onto a set of points (e.g. a line) in a parameter space $\mathcal{P}$. An area of the parameter space containing many mapped points (e.g. the intersection of many lines) indicates a potential feature of interest. In general, a linear segment s can be represented by its slope m_s and its axis intercept t_s in a system of Cartesian coordinates, i.e. $y = m_s \cdot x + t_s$. We can now take m and t as the axes of the parameter space and reformulate the line equation by $t_s = -m_s \cdot x + y$. Thus, each 2D picture point $p = (x_p, y_p) \in \mathcal{D}$ in the picture space is mapped on a line f_p with slope $-x_p$ and intercept y_p in the parameter space, i.e. a line f_p represented by $t = -m \cdot x_p + y_p$. The line f_p in the parameter space models all linear segments (lines) that pass through p in the original picture space. Thus, whenever several lines $f_{p_1}, \ldots, f_{p_k}$ in the parameter space intersect at a given point $(m_i, t_i) \in \mathcal{P}$, this indicates that the points $p_1, \ldots, p_k \in \mathcal{D}$ are located on a common line in picture space given by $y = m_i \cdot x + t_i$. A simplified example of the relationship between the picture space and the parameter space is visualized in Figure 14.1. The three picture points p_1, p_2, and p_3 are located on a common line s represented by $y = m_s \cdot x + t_s$ in the picture space (left). The corresponding mappings in the parameter space (right) f_{p_1}, f_{p_2}, and f_{p_3} intersect at point (m_s, t_s) in the parameter space.

Obviously, both the slope and the intercept are unbounded which may cause some problems when applying this basic technique. Thus, [45] proposed

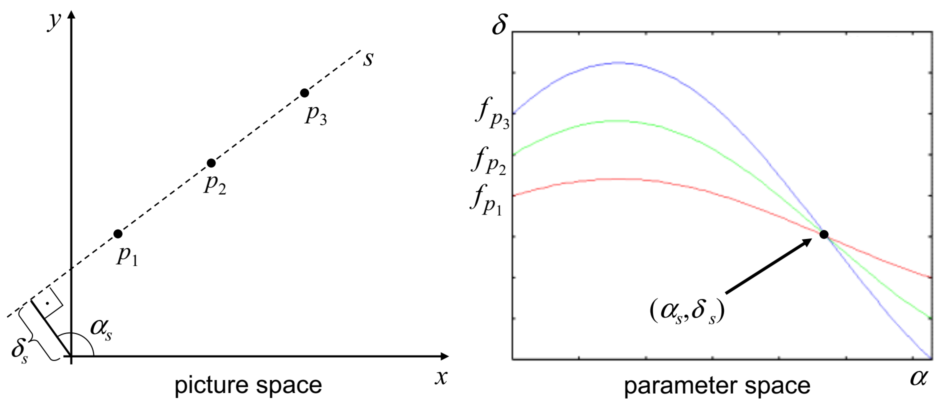


Figure 14.2: Hough transform from picture space to parameter space using angle and radius parameters.

to use spherical (also known as polar) coordinates, i.e. to use a parameter space based on angle and radius parameters rather than on slope and intercept parameters. The normal parametrization of a linear segment s in 2D is given by the angle α_s of its normal and its distance (radius) δ_s from the origin, i.e. s is represented by $x \cdot \cos\alpha_s + y \cdot \sin\alpha_s = \delta_s$. If α_s is restricted to the interval $[0, \pi)$, the normal representation of a line is unique. The mapping from the picture space onto the parameter space using angle/radius works similar to the mapping using slope/intercept. In either case, the parameter space represents all possible 1D lines in the original 2D data space.

In principle, each point of the data space is mapped on an infinite number of points in the parameter space which is not materialized as an infinite set but instead as a trigonometric function in the parameter space. Each function in the parameter space represents all lines in the picture space crossing the corresponding point in data space. The intersection of two curves in the parameter space indicates a line through both the corresponding points in the picture space. The objective of a clustering algorithm is to find intersections of many curves in the parameter space representing lines through many database objects. The key feature of the Hough transform is that the distance of the points in the original data space is not considered any more. Objects can be identified as associated to a common line even if they are far apart in the original feature space. As a consequence, the Hough transform is a promising candidate for developing a principle for subspace analysis that does not require the locality assumption and, thus, enables a global subspace clustering approach.

14.2 Subspace Analysis: a Novel Principle

Our novel principle for subspace analysis is based on a generalized description of spherical coordinates. Generalized spherical coordinates combine $d-1$ independent angles $\alpha_1, \ldots, \alpha_{d-1}$ with the norm r of a d-dimensional vector $x = (x_1, \ldots, x_d)^\mathsf{T}$ to completely describe the vector x w.r.t. the given orthonormal basis $e_1, \ldots, e_d$. We present a formalization analogously to [99]:

Definition 14.1 (Spherical coordinates)
Let e_i, $1 \leq i \leq d$, be an orthonormal basis in a d-dimensional feature space. Let $x = (x_1, \ldots, x_d)^\mathsf{T}$ be a d-dimensional vector on the hypersphere of radius r with center at the origin. Let u_i be the unit vector in the direction of the projection of vector x onto the manifold spanned by $e_i, \ldots, e_d$. For the $d-1$ independent angles $\alpha_1, \ldots, \alpha_{d-1}$, let α_i, $1 \leq i \leq d-1$, be the angle between u_i and e_i. Then the generalized spherical coordinates *of vector x are defined by:*

$$\begin{array}{rcl}
x_1 & = & r \cdot \cos(\alpha_1) \\
x_2 & = & r \cdot \sin(\alpha_1) \cdot \cos(\alpha_2) \\
 & \vdots & \\
x_i & = & r \cdot \sin(\alpha_1) \cdot \ldots \cdot \sin(\alpha_{i-1}) \cdot \cos(\alpha_i) \\
 & \vdots & \\
x_{d-1} & = & r \cdot \sin(\alpha_1) \cdot \ldots \cdot \sin(\alpha_{d-2}) \cdot \cos(\alpha_{d-1}) \\
x_d & = & r \cdot \sin(\alpha_1) \cdot \ldots \cdot \sin(\alpha_{d-2}) \cdot \sin(\alpha_{d-1})
\end{array}$$

Generally:

$$x_i = r \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i),$$

where $\alpha_d = 0$.

For any point $p \in \mathcal{D} \subseteq \mathbb{R}^d$ there exists an infinite number of hyperplanes containing p. The spherical coordinates are utilized to define the normal vector of the Hessian normal form for any of those hyperplanes, i.e., each hyperplane is uniquely defined by a point p and $d-1$ angles $\alpha_1, \ldots, \alpha_{d-1}$, with $\alpha_i \in [0, \pi)$, defining the normal vector. Thus, any point p together with any tuple of angles $\alpha_1, \ldots, \alpha_{d-1}$, can be mapped by the following *parametrization function* to the distance of the corresponding hyperplane to the origin.

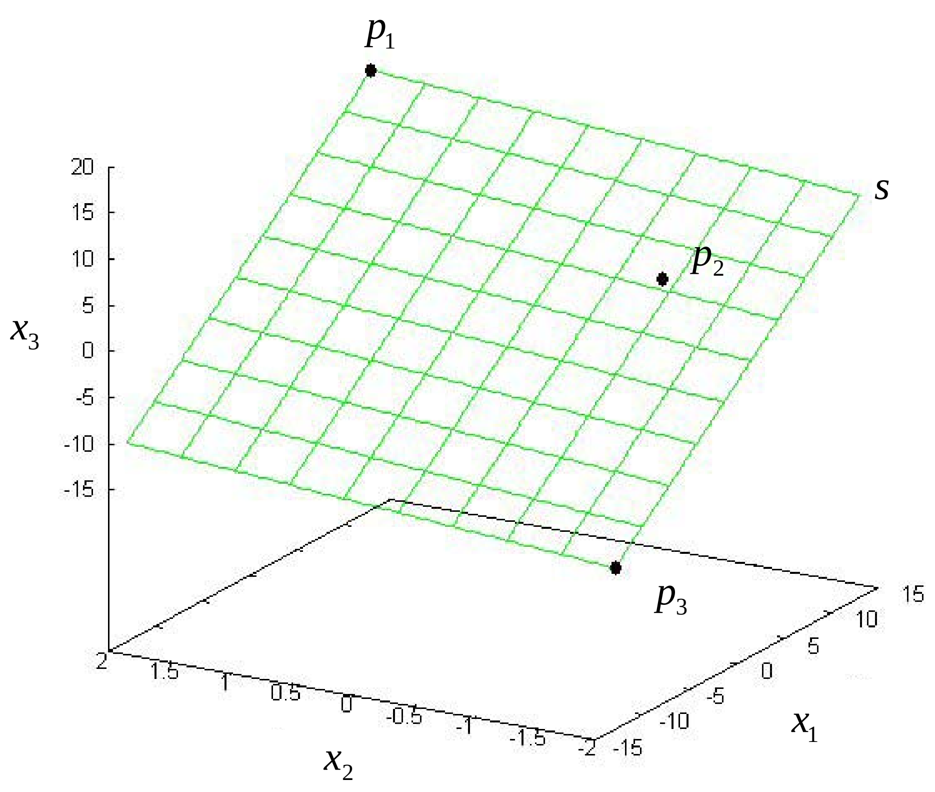
Definition 14.2 (Parametrization Function)
Let $p = (p_1, \ldots, p_d)^\mathsf{T} \in \mathcal{D} \subseteq \mathbb{R}^d$ *be a d-dimensional vector, and let* $n = (n_1, \ldots, n_d)^\mathsf{T}$ *be a d-dimensional unit vector specified by* $d-1$ *angles* $\alpha_1, \ldots, \alpha_{d-1}$ *according to Definition 14.1. Then the* parametrization function $f_p : \mathbb{R}^{d-1} \rightarrow \mathbb{R}$ *of vector p denotes the distance of the hyperplane defined by the point p and the normal vector n to the origin:*

$$\begin{aligned} f_p(\alpha_1, \ldots, \alpha_{d-1}) &= \langle p, n \rangle \\ &= \sum_{i=1}^{d} p_i \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i) \end{aligned}$$

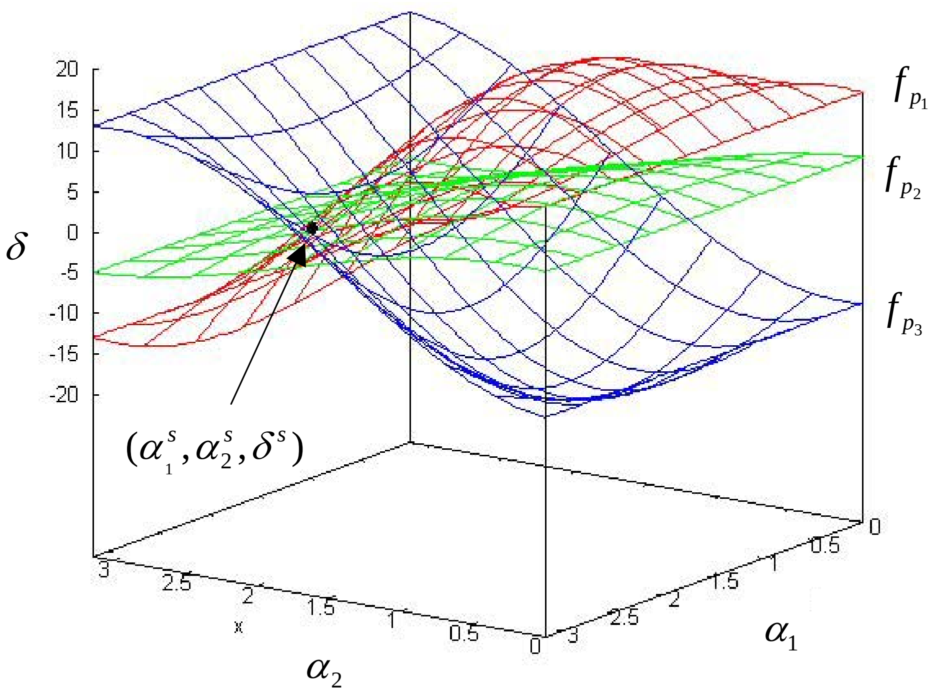
Based on Definition 14.2, we can map any point $p \in \mathbb{R}^d$ to a function in a d-dimensional parameter space $\mathcal{P}$ representing all possible hyperplanes containing p. This parameter space is spanned by the $d-1$ angles $\alpha_1, \ldots, \alpha_{d-1}$ of the normal vectors defining the hyperplanes in Hessian normal form and their distances $\delta = f_p(\alpha_1, \ldots, \alpha_{d-1})$ to the origin.

By means of the parametrization function (Definition 14.2), we can also extend the properties of the original Hough transform as stated in [45] for the mapping of 2-dimensional points to d-dimensional data spaces and the corresponding parameter spaces:

Property 14.1
A point $p \in \mathcal{D} \subseteq \mathbb{R}^d$ *in data space is represented by a sinusoidal curve* $f_p : \mathbb{R}^{d-1} \rightarrow \mathbb{R}$ *in parameter space* $\mathcal{P}$.



(a) Three points p_1, p_2, p_3 on a plane $s \subseteq \mathbb{R}^3$.



(b) Corresponding parametrization functions.

Figure 14.3: Transform of a 3-dimensional data space into a 3-dimensional parameter space.

Figure 14.3 illustrates a 3-dimensional example of this property. Three points p_1, p_2, and p_3 in data space are mapped onto the corresponding sinusoidal curves f_{p_1}, f_{p_2}, and f_{p_3}, respectively, in parameter space.

Property 14.2
A point $(\alpha_1, \ldots, \alpha_{d-1}, \delta) \in \mathcal{P}$ in parameter space corresponds to a $(d-1)$-dimensional hyperplane in data space.

In Figure 14.3, the point $(\alpha_1^s, \alpha_2^s, \delta^s)$ in parameter space represents the 2-dimensional plane s with

$$\delta^s = \cos(\alpha_1^s) \cdot x_1 + \sin(\alpha_1^s) \cdot \cos(\alpha_2^s) \cdot x_2 + \sin(\alpha_1^s) \cdot \sin(\alpha_2^s) \cdot x_3$$

in data space.

Property 14.3
Points that are located on a $(d-1)$-dimensional hyperplane in data space correspond to sinusoidal curves through a common point in parameter space.

The three points $p_1, p_2, p_3 \in \mathcal{D}$ (Figure 14.3) are located on the 2-dimensional plane s. Their corresponding sinusoidal curves $f_{p_1}, f_{p_2}, f_{p_3}$ intersect in the point $(\alpha_1^s, \alpha_2^s, \delta^s) \in \mathcal{P}$, where α_1^s, α_2^s and δ^s are the parameters of plane s as given above (cf. Property 14.2).

Property 14.4
Points located on the same sinusoidal curve in parameter space represent $(d-1)$-dimensional hyperplanes through the same point in data space.

For example, in Figure 14.3, f_{p_1} in parameter space represents all 2-dimensional planes through p_1 in data space. Thus, any point on f_{p_1} in parameter space represents a given 2-dimensional plane in data space that passes through p_1.

Properties 14.1 – 14.4 induce that an intersection point in the parameter space indicates points in the data space that are located on a common

$(d-1)$-dimensional hyperplane. In order to detect those linear hyperplanes in the data space, the task is to search for points in the parameter space where many sinusoidal curves intersect. Since computing all possibly interesting intersection points is computationally too expensive, we discretize the parameter space by some grid and search for grid cells with which many sinusoidal curves intersect. For that purpose, for each grid cell the number of intersecting sinusoidal curves is aggregated. Due to this discretization of the parameter space, exact intersections are no longer considered. Rather, a slight impreciseness is allowed modelling a certain degree of jitter given by the grid resolution. The higher the grid resolution is, the lower is the allowed degree of jitter, i.e. the more accurate the recognition of the line segments.

With the proposed concepts, we transform the original subspace clustering problem (in data space) into a grid-based clustering problem (in parameter space).

14.3 Specifying the Boundaries of the Grid

To define a discretization of the parameter space, the range of the axes must be known. The axes for the angle-parameters $\alpha_1, \ldots, \alpha_{d-1}$, are bounded by $[0, \pi)$. The δ-axis ranges from the minimum of all minima of all parametrization functions to the maximum of all their maxima within $[0, \pi)^{d-1}$. Each f_p is a sinusoid with a period of 2π. Thus, any f_p has exactly one global extremum in the interval $[0, \pi)^{d-1}$. If the extremum of f_p is a maximum, the minimal value for f_p in the given interval has to be determined and *vice versa*.

To find the global extremum of a parametrization function f_p in the interval $[0, \pi)^{d-1}$, those angles $\alpha_1, \ldots, \alpha_{d-1}$ need to be determined where all the first order derivatives of f_p are zero, and the Hessian matrix of f_p is either positive or negative definite. As noted above, f_p is guaranteed to have exactly one global extremum $f_p(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{d-1})$ in $[0, \pi)^{d-1}$. The values for the angles $\tilde{\alpha}_n$ $(n = 1, \ldots, d-1)$ of the global extremum of f_p are given by (cf.

Section 14.4.1 for details):

$$\tilde{\alpha}_n = \arctan\left(\frac{\sum_{j=n+1}^{d} p_j \cdot \left[\prod_{k=n+1}^{j-1} \sin(\tilde{\alpha}_k)\right] \cdot \cos(\tilde{\alpha}_j)}{p_n}\right)$$

Given the global extremum of a parametrization function f_p in the interval $[0,\pi)^{d-1}$, we have to distinguish several cases to determine the opposite value, i.e., to determine the maximum of f_p if the global extremum of f_p is a minimum, or, to determine the minimum of f_p if the global extremum is a maximum. In the following, we describe how to determine the point $\alpha^{\min}$ where the parametrization function f_p has a minimum in interval $[0,\pi)^{d-1}$ given that the global extremum is a maximum. In the opposite case, the point $\alpha^{\max}$ where the parametrization function f_p has a maximum in interval $[0,\pi)^{d-1}$ given the global extremum is a minimum can be determined analogously. Please refer to Section 14.4.2 for a detailed formalization of this step.

We determine the point $\alpha^{\min} = (\alpha_1^{\min}, \dots, \alpha_{d-1}^{\min})$ where the parametrization function f_p has a minimum in interval $[0,\pi)^{d-1}$ as follows: First, the angle $\overline{\alpha}_{d-1}$ on axis $(d-1)$ is determined where f_p has an extremum on this axis. Dependent on the type of the extremum in $\overline{\alpha}_{d-1}$ and the location of $\overline{\alpha}_{d-1}$ in the interval $[0,\pi)$, the minimum angle $\alpha_{d-1}^{\min}$ on axis $(d-1)$ in interval $[0,\pi)$ is determined. In the next step, axis $(d-2)$ will be considered: Now, the angle $\overline{\alpha}_{d-2}$ will be determined, where f_p has an extremum on this axis under the constraint of the known minimum on axis $d-1$, which is given by $\alpha_{d-1}^{\min}$. Analogously to the first step, dependent on the type of the extremum in $\overline{\alpha}_{d-2}$ and the location of $\overline{\alpha}_{d-2}$ in the interval $[\check{\alpha}_{d-2}, \hat{\alpha}_{d-2})$, the minimum angle $\alpha_{d-2}^{\min}$ is determined. In this way, all minimum angles are determined under the constraint of the known minima on the already processed axes.

In summary, given for each parametrization function f_p its minimal and maximal value $\alpha_p^{\min}$ and $\alpha_p^{\max}$ in interval $[0,\pi)^{d-1}$, the δ-axis of the parameter space $\mathcal{P}$ is bounded by

$$[\delta_{min}, \delta_{max}] = [\min_{p\in\mathcal{D}}(f_p(\alpha_p^{\min})), \max_{p\in\mathcal{D}}(f_p(\alpha_p^{\max}))]$$

and $\mathcal{P} = [\delta_{min}, \delta_{max}] \times [0, \pi)^{d-1}$.

14.4 Finding the Extrema of the Parametrization Functions

In the following, a detailed formalization of the computational steps for identifying the extrema of a parametrization function f_p are given. First (14.4.1), the determination of the global extremum of a parametrization function f_p in the interval $[0, \pi)^{d-1}$ is described. Then (14.4.2), based on this derivation, it is specified, how to identify the minimum of f_p in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$. The maximum of f_p in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$ can be determined analogously.

14.4.1 Global Extremum

Each parametrization function f_p is a sinusoid with a period of 2π. Thus, any f_p has exactly one global extremum in the interval $[0, \pi)^{d-1}$. To find the global extremum of f_p, those angles $\alpha_1, \ldots, \alpha_{d-1}$ need to be determined where all the first order derivatives of f_p are zero, and the Hessian matrix is either positive or negative definite. The *first order partial derivatives* of the parametrization function f_p are given by:

$$\frac{\partial f_p}{\partial \alpha_n}(\alpha) = \prod_{i=1}^{n-1} \sin(\alpha_i) \cdot \left(-p_n \cdot \sin(\alpha_n) + \sum_{j=n+1}^{d} p_j \cdot \cos(\alpha_n) \cdot \left[\prod_{k=n+1}^{j-1} \sin(\alpha_k) \right] \cdot \cos(\alpha_j) \right)$$

Accordingly, the *Hessian Matrix of* f_p is defined as

$$H_{f_p}(\alpha) \quad = \quad \left(\frac{\partial^2 f_p}{\partial \alpha_n \partial \alpha_m} \right)(\alpha) \in \mathbb{R}^{d-1 \times d-1}$$

for $1 \leq n, m \leq d-1$.

The extrema of parametrization function f_p are characterized by the following properties:

1. $\tilde{\alpha} = (\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{d-1})$ is an extremum point of $f_p \Rightarrow \nabla f_p(\tilde{\alpha}) = 0$, i.e.
$$\frac{\partial f_p}{\partial \alpha_1}(\tilde{\alpha}) = \ldots = \frac{\partial f_p}{\partial \alpha_{d-1}}(\tilde{\alpha}) = 0.$$
2. $\nabla f_p(\tilde{\alpha}) = 0$ and the Hessian matrix H_{f_p} at $\tilde{\alpha}$ is positive definite $\Rightarrow \tilde{\alpha}$ is a minimum point
3. $\nabla f_p(\tilde{\alpha}) = 0$ and the Hessian matrix H_{f_p} at $\tilde{\alpha}$ is negative definite $\Rightarrow \tilde{\alpha}$ is a maximum point

For any first order partial derivative $\frac{\partial f_p}{\partial \alpha_n}(\tilde{\alpha}) \doteq 0$, $(1 \leq n \leq d-1)$, one of the following conditions holds:

$$\begin{aligned}
\sin(\tilde{\alpha}_1) &= 0 \\
&\vdots \\
\sin(\tilde{\alpha}_{n-1}) &= 0 \\
\tan(\tilde{\alpha}_n) &= \frac{\sum\limits_{j=n+1}^{d} p_j \cdot \left[\prod\limits_{k=n+1}^{j-1} \sin(\tilde{\alpha}_k)\right] \cdot \cos(\tilde{\alpha}_j)}{p_n}
\end{aligned}$$

Since the first $n-1$ conditions yield an indefinite Hessian matrix, according to the last condition, a point $\tilde{\alpha} = (\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{d-1})$ can be an extremum point of parametrization function f_p only if

$$\tilde{\alpha}_n = \arctan\left(\frac{\sum\limits_{j=n+1}^{d} p_j \cdot \left[\prod\limits_{k=n+1}^{j-1} \sin(\tilde{\alpha}_k)\right] \cdot \cos(\tilde{\alpha}_j)}{p_n}\right)$$

As noted above, f_p is guaranteed to have exactly one global extremum $f_p(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{d-1})$ in $[0, \pi)^{d-1}$. The values for the angles $\tilde{\alpha}_n, n = 1, \ldots, d-1$ of the global extremum are given by the equation above.

14.4.2 Minimum and Maximum Value

Let $\check{\alpha} = (\check{\alpha}_1, \ldots, \check{\alpha}_{d-1})$ and $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_{d-1})$ for a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$, $1 \leq i \leq d-1$. To determine the point $\alpha^{min} = (\alpha_1^{min}, \ldots, \alpha_{d-1}^{min})$

where the parametrization function f_p has a minimum in interval $[\check{\alpha}, \hat{\alpha})$ the following steps for each dimension $n = d-1, \ldots, 1$ have to be performed:

1. Let
$$\overline{\alpha}_n = \arctan\left(\frac{\sum\limits_{j=n+1}^{d} p_j \cdot \left[\prod\limits_{k=n+1}^{j-1} \sin(\alpha_k^{\min})\right] \cdot \cos(\alpha_j^{\min})}{p_n}\right)$$
be the value where f_p has an extremum on the n-th axis under the constraint of known minimum angles $\alpha_{n+1}^{\min}, \ldots, \alpha_{d-1}^{\min}$.

2. Given $\alpha = (c_1, \ldots, c_{n-1}, \overline{\alpha}_n, \alpha_{n+1}^{\min}, \ldots, \alpha_{d-1}^{\min}) \in [\check{\alpha}, \hat{\alpha})^{n-1} \times [0, \pi) \times [\check{\alpha}, \hat{\alpha})^{d-1-n}$, where c_i are arbitrarily chosen values in $[\check{\alpha}, \hat{\alpha})$, we differentiate the following cases:

   i. f_p has a *maximum* in α:
      - A. $\check{\alpha}_n \le \overline{\alpha}_n \le \hat{\alpha}_n$:
        - A1. $\overline{\alpha}_n - \check{\alpha}_n \le \hat{\alpha}_n - \overline{\alpha}_n$: $\alpha_n^{\min} \to \hat{\alpha}_n$.
        - A2. $\overline{\alpha}_n - \check{\alpha}_n > \hat{\alpha}_n - \overline{\alpha}_n$: $\alpha_n^{\min} = \check{\alpha}_n$.
      - B. $\overline{\alpha}_n < \check{\alpha}_n$: $\alpha_n^{\min} \to \hat{\alpha}_n$.
      - C. $\overline{\alpha}_n > \hat{\alpha}_n$: $\alpha_n^{\min} = \check{\alpha}_n$.

      As illustrated in Figure 14.4, if $\overline{\alpha}_n$ is inside the interval and nearer to the left boundary (A1), the minimum value $\alpha_n^{\min}$ is located at the right boundary and *vice versa* (A2). If $\overline{\alpha}_n$ is outside the interval (B and C), the minimum value $\alpha_n^{\min}$ is located at the opposite boundary.

   ii. f_p has a *minimum* in α: The same principle of reasoning has to be applied contrariwise.
      - A. $\check{\alpha}_n \le \overline{\alpha}_n \le \hat{\alpha}_n$: $\alpha_n^{\min} = \overline{\alpha}_n$.
      - B. $\overline{\alpha}_n < \check{\alpha}_n$: $\alpha_n^{\min} = \check{\alpha}_n$.
      - C. $\overline{\alpha}_n > \hat{\alpha}_n$: $\alpha_n^{\min} \to \hat{\alpha}_n$.

The maximum $\alpha^{max} = (\alpha_1^{max}, \ldots, \alpha_{d-1}^{max})$ of f_p in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$ can be determined analogously.

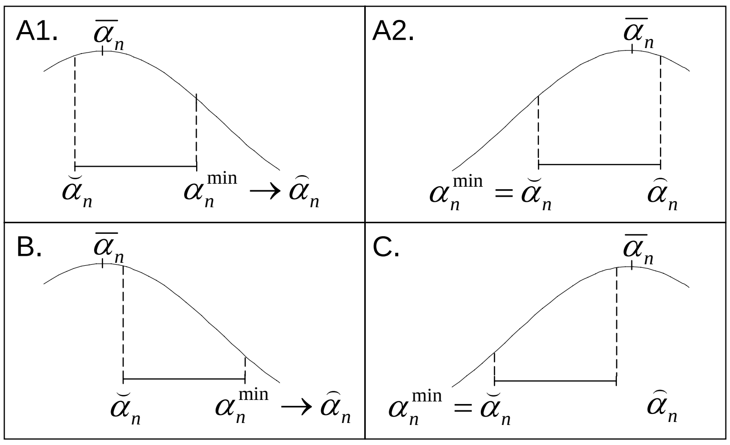
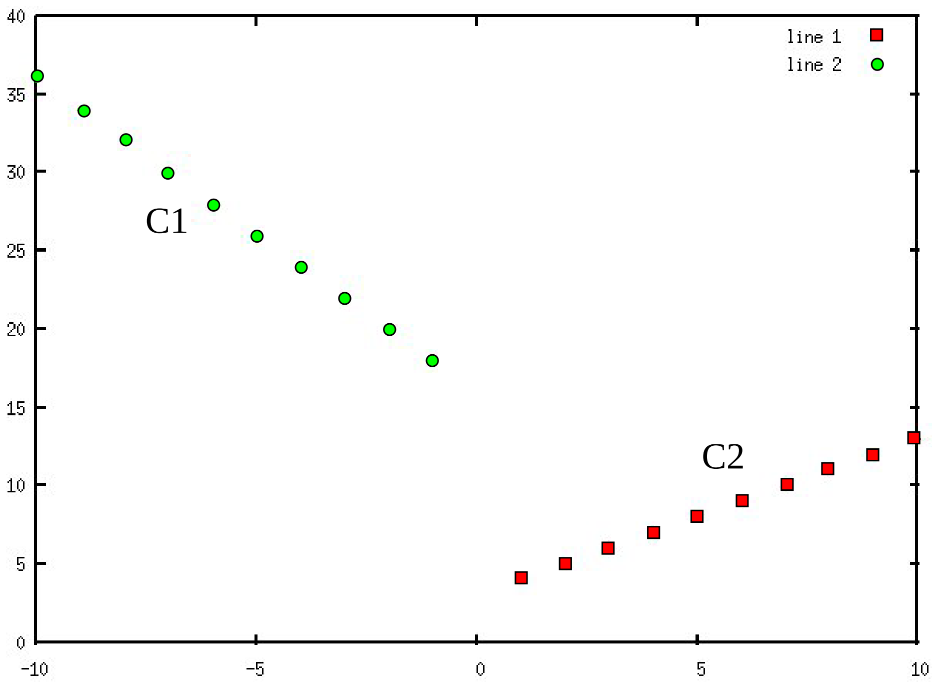

A1.
$\bar{\alpha}_n$
$\breve{\alpha}_n$
$\alpha_n^{\min} \rightarrow \hat{\alpha}_n$
A2.
$\bar{\alpha}_n$
$\alpha_n^{\min} = \breve{\alpha}_n$
$\hat{\alpha}_n$
B.
$\bar{\alpha}_n$
$\breve{\alpha}_n$
$\alpha_n^{\min} \rightarrow \hat{\alpha}_n$
C.
$\bar{\alpha}_n$
$\alpha_n^{\min} = \breve{\alpha}_n$
$\hat{\alpha}_n$

Figure 14.4: Different cases for finding the minimum of a parametrization function in a given interval.

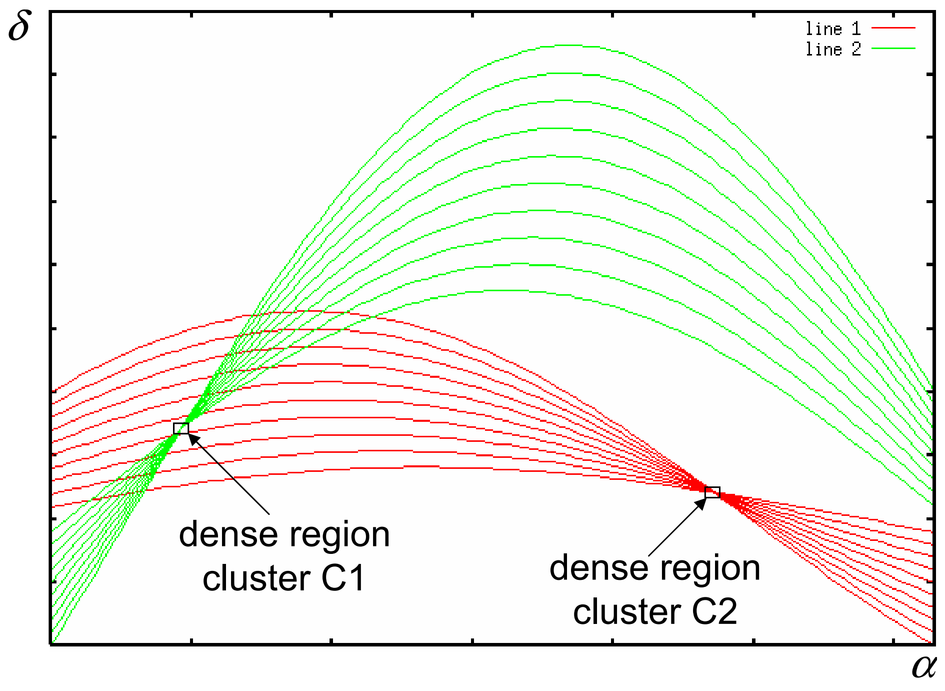
14.5 Identifying Dense Grid Cells

Given a discretized parameter space, now those grid cells (hypercuboids) have to be found that are intersected by parametrization functions of a minimum number μ of functions. Hypercuboids containing at least μ parametrization functions are called dense regions of the parameter space. Those dense regions represent arbitrarily oriented subspaces in the data space accommodating at least μ points. This is illustrated in Figure 14.5. The two subspace clusters forming lines in the data space (cf. Figure 14.5(a)) are represented by two distinct dense regions in the parameter space (cf. Figure 14.5(b)).

To find those dense regions in the parameter space, for each grid cell or hypercuboid the number of parametrization functions which intersect this hypercuboid has to be counted. This can be done conveniently by determining those values $\alpha_p^{\min}$ and $\alpha_p^{\max}$ in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$ that minimize and maximize a parametrization function f_p. Then, all hypercuboids based on this interval and positioned between $f_p(\alpha_p^{\min})$ and $f_p(\alpha_p^{\max})$ are intersected by f_p. The values $\alpha_p^{\min}$ and $\alpha_p^{\max}$ in a given interval $[\check{\alpha}, \hat{\alpha}) \subseteq [0, \pi)^{d-1}$ that minimize and maximize f_p can be determined analogously to the algorithm specified in Section 14.3 where the given interval was assumed to be $[0, \pi)^{d-1}$.



(a) Two lines in data space.



(b) Dense regions in parameter space.

Figure 14.5: Dense regions in parameter space capturing two lines in data space.

14.6 Efficiently Finding Regions of Interest

A region qualifying as a dense region, but containing exclusively one cluster, possibly need to be defined by a rather small interval of angles and also a rather small interval of distances from the origin because otherwise the same interval could also contain functions representing points of other clusters (cf. the dense region of cluster C1 in Figure 14.5). For that purpose, a rather high number of intervals in each dimension of the parameter space is needed, resulting in a huge number of grid cells possibly qualifying as dense regions. Thus, searching the parameter space with a predefined grid in the range $[0, \pi)$ for each angle and $[\delta_{min}, \delta_{max}]$ for the distance from the origin, is not feasible for high dimensional data in terms of space and time complexity.

To avoid exponential complexity, the following search strategy for the parameter space is proposed:

Step 1 The axes (distance and angles) are divided successively in a static order given by $\delta, \alpha_1, \ldots, \alpha_{d-1}$. After dividing one axis, from the resulting 2 hypercuboids the one containing most points is selected for refinement. If both hypercuboids contain an equal amount of points, the first one is selected (arbitrarily). The selected hypercuboid is divided recursively by splitting the next axis. The neglected hypercuboid is kept in a queue.

Step 2 If both children of a divided hypercuboid contain less than μ points, the search in the corresponding path is discontinued. Unless the queue is empty, the next hypercuboid in the queue is examined using the same procedure. In the queue, hypercuboids are ordered descendingly by the amount of points contained by a hypercuboid. If two hypercuboids contain an equal amount of points, the smaller one is preferred, since a smaller interval containing an equal amount of data points is a more promising candidate.

Step 3 At a predefined depth (i.e. a given number s of successive splits), a hypercuboid (i.e. the corresponding interval) is considered to be sufficiently small to define a hyperplane containing a subspace cluster. If the number of points within the hypercuboid exceeds a predefined number μ of points, these points are considered to build a subspace cluster. The corresponding subspace is treated as a new data space containing all the points accounted for in the hypercuboid. This new data space of dimensionality $d - 1$ undergoes the same procedure recursively, while $d > 2$, i.e., CASH is called for the points in the hypercuboid using the corresponding subspace as data space (see Section 14.7 for a more detailed explanation of the recursive descent). If no subspace cluster of lower dimensionality is found in this $(d - 1)$-dimensional space, all the points in this subspace are supposed to build a $(d-1)$-dimensional subspace cluster.

Step 4 All points participating at a $(d-1)$-dimensional subspace cluster derived at a search path are removed from the d-dimensional data space. The queue is reorganized and hypercuboids are removed, if they contain now less than μ points. A new search path based on the next hypercuboid in the queue is pursued.

Step 5 The search is complete, if in the d-dimensional space no interval is found containing at least μ points.

This search strategy determines clusters of at least μ points in any arbitrarily oriented subspace and provides a description with an accuracy regarding the orientation α and the distance δ from the origin as defined by the predefined number s of splits.

Unlike traditional grid-based clustering approaches, CASH has no problems if a region of interest (i.e., a cluster) is located at the boundary of two connected grid cells, g_1 and g_2. In that case, the functions will intersect both neighboring grid cells and both grid cells, g_1 and g_2, will be dense. CASH will refine one of these grid cells (e.g. g_1 – cf. step 1) until the cluster is found. After that, CASH eliminates the participating points (i.e., functions) and, thus, the second grid-cell (g_2) will not be dense anymore (step 4).

Due to the recursive search in an obtained cluster (step 3), a cluster hierarchy is gained along the way, i.e., a subspace cluster may in turn contain nested subspace clusters of lower dimensionality. In that case, it may be interesting to report all nested clusters and the information of the "contained-in" relationships. Section 14.8 provides more details on how a such a hierarchy can be obtained.

14.7 Recursive Descent

In this Section, we describe in more detail the recursive procedure to find lower dimensional clusters within higher dimensional clusters as mentioned in Step 3 of the search heuristic (Section 14.6).

If CASH finds a cluster, i.e. a d-dimensional hypercuboid g ($d > 2$) at a predefined depth (i.e., a given number s of successive splits) being sufficiently dense, the search space is transformed according to the current orientation and affinity of the subspace defined by the hypercuboid g. In particular, the hypercuboid g defines a subspace by means of the Hessian normal form with a certain error (as defined by the intervals of angles $[\check{\alpha}_i, \hat{\alpha}_i)$ for each axis i and the interval of distances $[\delta_{min}, \delta_{max}]$ from the origin spanned by g). The corresponding hyperplane (a $(d-1)$-dimensional affine subspace) is given by spherical coordinates of the normal vector n assuming the mean of δ_{min} and δ_{max} as the length (radius r) of n and for each angle the mean of the corresponding values of $\check{\alpha}$ and $\hat{\alpha}$, respectively. In other words, the spherical coordinates of n are given by

$$r = \frac{\delta_{min} + \delta_{max}}{2} \tag{14.1}$$

and

$$\alpha_i = \frac{\check{\alpha}_i + \hat{\alpha}_i}{2} \tag{14.2}$$

(for $1 \leq i \leq d-1$). The Cartesian coordinates are given as in Definition 14.1 by

$$x_i = r \cdot \left(\prod_{j=1}^{i-1} \sin(\alpha_j) \right) \cdot \cos(\alpha_i). \tag{14.3}$$

The normal vector n is then completed to an orthonormal basis by adding $d-1$ linear independent arbitrary basis vectors (which is generally possible in a d-dimensional space). The corresponding orthonormal matrix $\boldsymbol{N}$ facilitates the transformation of the parametrization functions from the d-dimensional space into the $(d-1)$-dimensional subspace by multiplication with $\boldsymbol{N}$ and projection onto the space given by $\boldsymbol{N}\backslash n$. This way, a new, $(d-1)$-dimensional subspace is defined. The data set corresponding to this subspace contains only the parametrization functions intersecting the hypercuboid g. For the next step, CASH is applied to the points of the cluster represented by g transformed into the new $(d-1)$-dimensional subspace. In each next step the search space is therefore reduced in dimensionality and at least not increased w.r.t. the number of database objects.

14.8 Deriving a Hierarchy of Subspace Clusters

By means of the recursive descent, CASH directly yields a hierarchy of arbitrarily oriented subspace clusters. All points belonging to a dense $(d-1)$-dimensional grid cell also belong to the d-dimensional grid cell that has been previously analyzed in order to find lower dimensional clusters. Thus, when recursively descending after identifying a cluster, we simply have to store a pointer from the higher dimensional cluster to the lower dimensional cluster. As a result, we get a containment hierarchy of clusters and their corresponding subspaces. This hierarchy displays an important relationship among clusters. If any l-dimensional cluster A is contained in a k-dimensional cluster B $(l < k)$ according to this relationship, this means that all points of cluster A are not only located on a common l-dimensional cluster hyperplane but also located on the k-dimensional cluster hyperplane that is shared by the points in B. Cluster B can thus be regarded as a superset of A. A higher dimensional superset B of a cluster A can be regarded as an interesting cluster itself, if $|B - A| \geq \mu$.

From the point of view of a hierarchy of subspaces, the difference between the points in $B - A$ and the points in B is that points in $B - A$ exhibit a correlation not only among the l attributes that are correlated for the points in B but also among $k-l$ additional attributes. Knowing these relationships is quite interesting when evaluating and interpreting the reported clusters in order to find hidden causalities in the data.

The hierarchy can be visualized as a tree. Each node of a tree represents a cluster. The root node (level 0) of the tree represents the entire database forming a "dummy" d-dimensional cluster in which all other "true" clusters are contained. A node at level k represents a $(d-k)$-dimensional cluster. An edge between a k- and an l-dimensional cluster ($l < k$) represents the containment of the l-dimensional cluster within the k-dimensional one. Finally, any node on level $l \geq 1$ without parent node is linked to the root.

14.9 Properties of the Algorithm

The algorithm CASH transforms the data objects from $\mathcal{D} \subseteq \mathbb{R}^d$ into a corresponding parameter space (based on radius and angles) $\mathcal{P} = [\delta_{min}, \delta_{max}] \times [0, \pi)^{d-1}$. After that, CASH identifies dense regions in that parameter space using the search strategy proposed above. These dense regions represent arbitrarily oriented subspace clusters in the data space. For each dense region, a recursive descent is initialized. The resulting hierarchy of subspace clusters is visualized by a tree structure placing the complete database in the root of the tree representing the entire database.

14.9.1 Complexity

Let N be the number of data points in a d dimensional data space. When bisecting the parameter space of α_i and δ, we need to determine those database points, whose parameter functions intersect with the generated cells in the parameter space. This is done by the maximization and minimization of δ given the constraints on α_i (i.e., $\check{\alpha}_i \leq \alpha_i < \hat{\alpha}_i$ for all $1 \leq i \leq d-1$) requiring $O(d^3)$ time per object and cell.

The CASH algorithm performs a recursive bisection of the data space where all bisections with fewer than μ associated database points are discarded. Since bisection cells which do not belong to any cluster are only randomly associated to a few arbitrary points, the bisection process for those cells stops at a high level of the bisection tree. Only cells belonging to actual subspace clusters are bisected until the defined maximum number s of bisection levels is reached. Therefore, for a data set containing $c > 0$ clusters, a number $O(s \cdot c)$ of nodes in the bisection tree are encountered, each causing $O(N \cdot d^3)$ work to find all subspace clusters. Together, we have an average time complexity in $O(s \cdot c \cdot N \cdot d^3)$.

14.9.2 Input Parameters

CASH requires the user to specify two input parameters: The first parameter μ specifies the minimum number of sinusoidal curves that need to intersect a hypercuboid in the parameter space such that this hypercuboid is regarded as a dense area. Obviously, this parameter represents the minimum number of points in a cluster and thus is very intuitive. The second parameter s specifies the maximal number of splits along a search path (splitlevel). Thus, it controls the maximal allowed deviation from the hyperplane of the cluster in terms of orientation and jitter. We show in our experiments, that CASH is rather robust w.r.t. s. Since CASH does not require parameters that are hard to guess like the number of clusters, the average dimensionality of the subspace clusters, or the size of the Euclidean neighborhood based on which the similarity of the subspace clusters is learned, it is much more usable and stable than its competitors.

14.9.3 Alternative Parametrization

It is also possible to treat the splitlevel for the radius and the angles separately. This increases the number of parameters by one but allows to treat different kinds of deviations from the idealized cluster hyperplane differently: The allowed variance in the radius corresponds to the allowed thickness of the hyperplane, i.e., the tolerated deviation of cluster members orthogonally

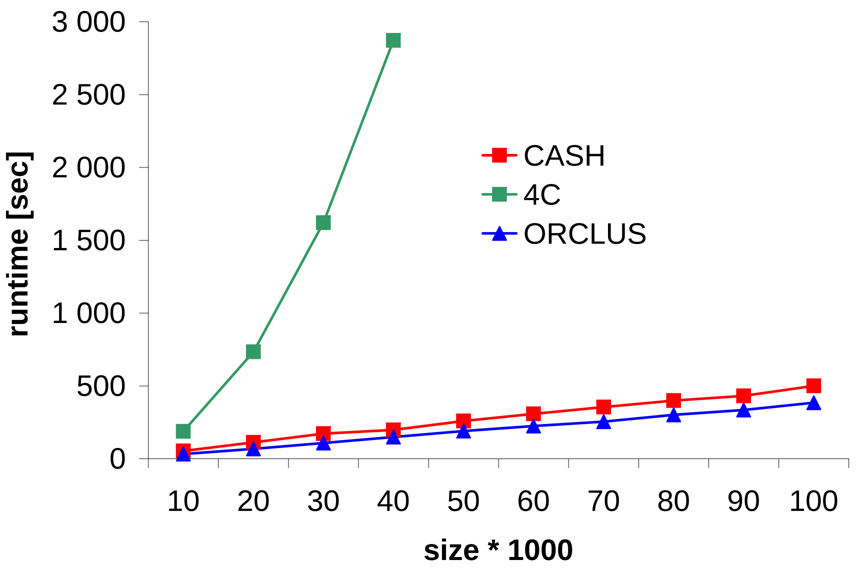


Figure 14.6: Scalability w.r.t. size.

from the hyperplane. This kind of error is usually encountered in real world data sets. The tolerated variance in the angles corresponds to the tolerated variance in the orientation of the hyperplane. A larger allowance here makes it possible to grasp clusters where the members do not follow a perfectly linear correlation.

14.10 Evaluation

14.10.1 Efficiency

To evaluate the scalability of CASH w.r.t. the size of the data set, we created ten data sets containing four, equally sized one dimensional clusters in a 5 dimensional data space with an increasing number of points ranging from 10,000 to 100,000. CASH performs comparably well to ORCLUS. Both outperform 4C significantly (cf. Figure 14.6). As a fair setting, we gave as parameter k to ORCLUS the exact number of clusters in the data set (i.e. $k = 4$), and parameter l has been set to the correct correlation dimensionality of the clusters (i.e. $l = 1$). For 4C, the parameters have been set to

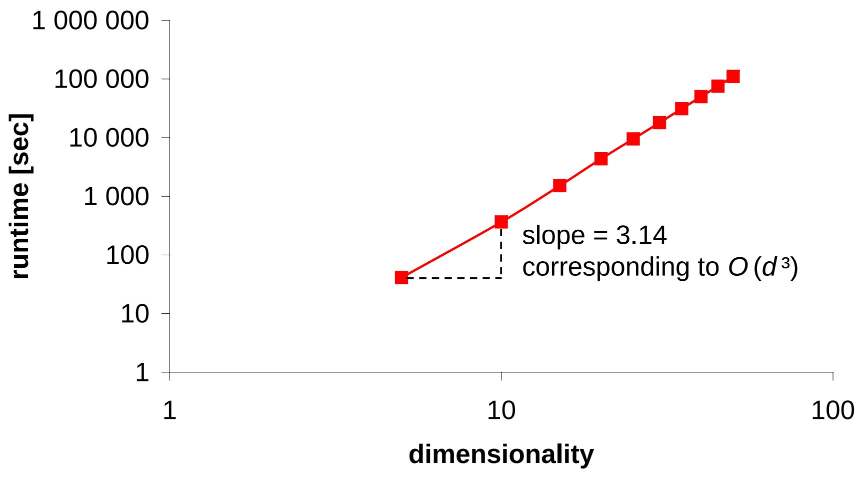


Figure 14.7: Scalability of CASH w.r.t. dimensionality.

MinPts = 100, $\varepsilon = 0.1$, $\lambda = 1$, and $\delta = 0.01$, reflecting the actual cluster structure in the synthetic data sets. The parameter setting for CASH was $s = 40$ and $\mu = 2,500$.

To assess the impact of the dimensionality of the data space on the runtime of CASH, we created 10 data sets ranging in dimensionality from 5 to 50, each data set containing a one dimensional cluster of 10.000 points. The parameters were set to $s = 50$ and $\mu = 5,000$. Figure 14.7 shows the scalability of CASH logarithmically on both axes, dimensionality and runtime. The graph is a line with slope 3.14, approximately corresponding to the expected runtime behavior.

In both test scenarios, the objective was to find 1-dimensional clusters in a d-dimensional data space, since this is the most complex task for CASH, requiring a maximal recursive descent from $d-1$ until subspace dimensionality 1 is reached.

14.10.2 Effectiveness

The parameter s clearly influences the runtime behavior to a certain degree. However, CASH reaches satisfying behavior in terms of effectiveness for even

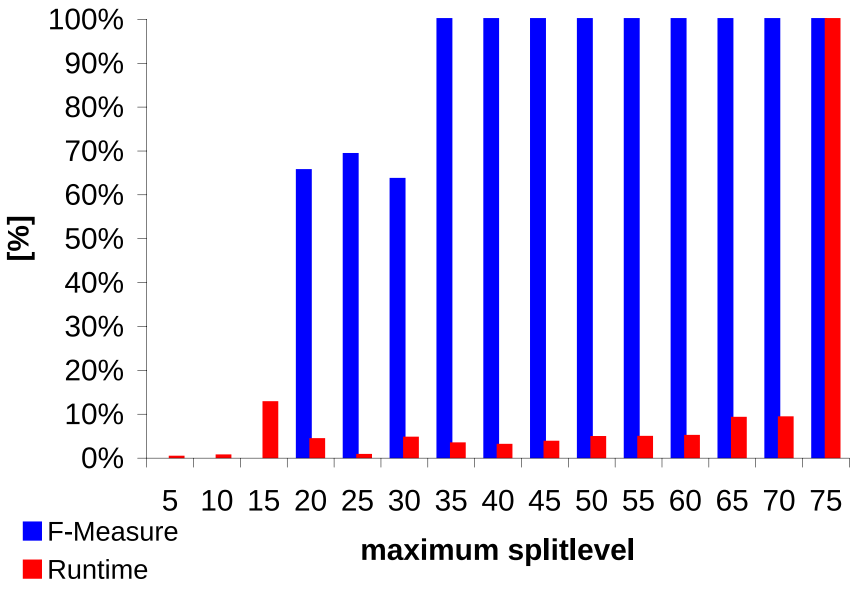


Figure 14.8: F-measure and runtime of CASH w.r.t. maximum split level.

relatively low values for s. Figure 14.8 illustrates the effect of s on runtime and effectiveness simultaneously. On a 5-dimensional data set containing two 1-dimensional clusters, each containing 500 points, and 500 points of noise, CASH reaches an F-measure of 100% already for $s = 35$, while the runtime remains relatively low with 3.29% compared to the maximum runtime for $s = 75$.

To assess the robustness of CASH against noise, we created ten data sets containing an increasing level of noise objects ranging from 0 to 90% of the complete data set. Figure 14.9 shows the comparison in robustness with ORCLUS and 4C. The parameter setting for ORCLUS has been $l = 1$ and $k = 2$, reflecting the true number of clusters and their dimensionality. For 4C, the optimal parameter setting has been used with $MinPts = 5$, $\varepsilon = 0.12$, $\lambda = 1$, and $\delta = 0.01$. For CASH, the parameters have been chosen as $s = 30$ and $\mu = 50$. Let us note that CASH did not require any efforts for optimization of parameter settings. While both 4C and ORCLUS performed relatively well for very low levels of noise objects, their performance deteriorates for a higher degree of noise. CASH remains constantly on an F-measure of 100% up to a noise level of 80%. Even for an extremely high level of noise (90%),

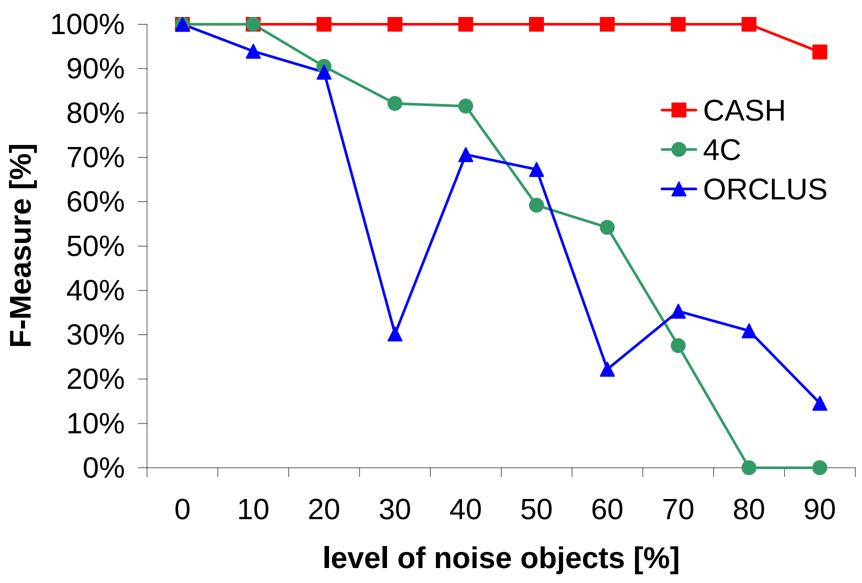
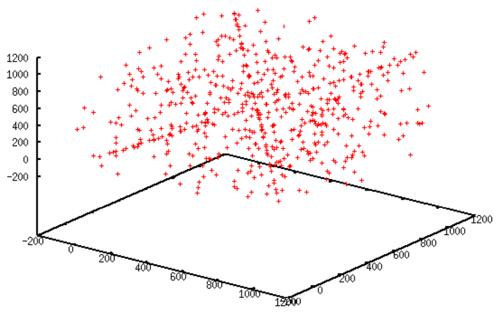


Figure 14.9: F-Measure w.r.t. noise level.

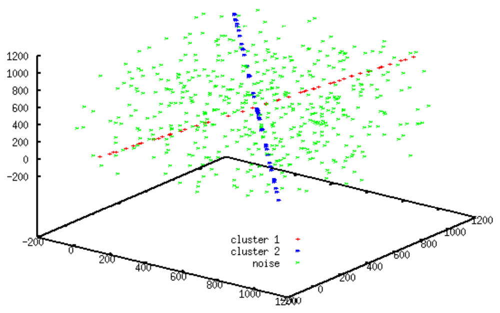
CASH still reaches an F-measure of 94%.

We illustrate the robustness of CASH against noise on an exemplary 3-dimensional data set depicted in Figure 14.10(a). CASH finds the 2 1-dimensional subspace clusters (each of size 50) embedded in 500 noise points exactly (cf. Figure 14.10(b)). The results of ORCLUS (with optimal parameter setting $l = 1$ and $k = 2$) are shown in Figures 14.10(c) and 14.10(d). As it can be seen, the clusters found by ORCLUS do not reflect the real cluster structure at all. For 4C, we tried several parameter settings. Unfortunately, 4C was never able to find a meaningful cluster structure at all.

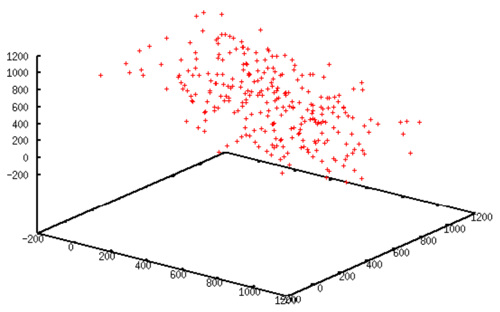
Further experiments on high dimensional data sets have been performed with CASH, 4C, and ORCLUS. The data sets contained complex subspace cluster structures with sparse clusters, including subspace clusters of significantly differing dimensionality, subspace clusters hierarchically embedded in higher dimensional subspaces, and noise objects. In none of the performed experiments, 4C or ORCLUS were able to find meaningful clusters, while CASH exactly detected the cluster structures in most cases. As an example, we present the results on a complex 3-dimensional data set shown in Figure 14.11(a), containing three 1-dimensional clusters each of 500 points, two 2-



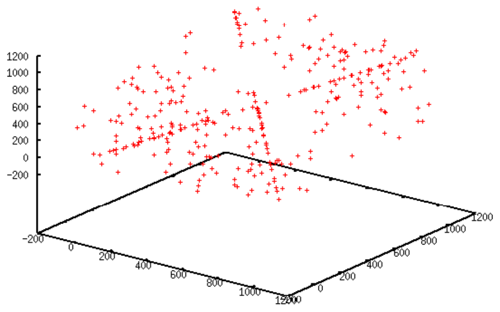
(a) Synthetic data set DS1.



(b) CASH - Clustering.

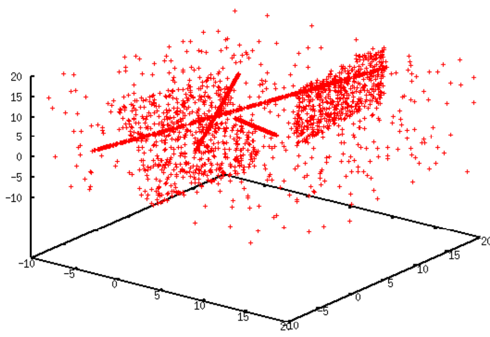


(c) ORCLUS - Cluster 1.

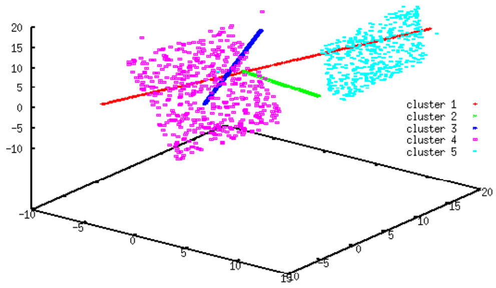


(d) ORCLUS - Cluster 2.

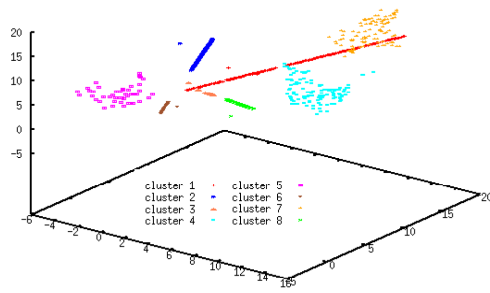
Figure 14.10: Clustering synthetic data set DS1.



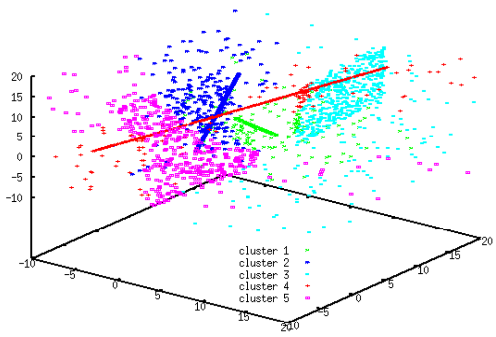
(a) Data set DS2.



(b) CASH – Cluster 1 - 5.

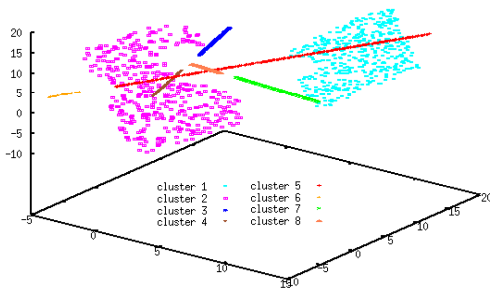


(c) 4C – Cluster 1 - 8.

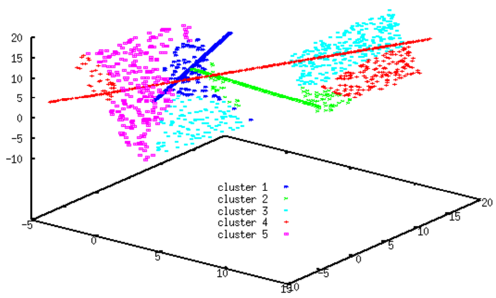


(d) ORCLUS – Cluster 1 - 5.

Figure 14.11: Clustering results on synthetic data set DS2.



(a) 4C – Cluster 1 - 8.



(b) ORCLUS – Cluster 1 - 5.

Figure 14.12: Clustering results on DS2 after noise removal.

Table 14.1: CASH clustering on Wages data.

c ID	dim	# objects	Description
1	2	215	YE = 12; A - YW = 18
2	2	70	YE = 16; A - YW = 22
3	3	247	YE + YW = A - 6

dimensional planes each containing 500 points, and 500 points of noise. One of the planes is intersected by two lines, the other plane is intersected by one line. Here, CASH is able to identify the cluster structure of all 5 clusters exactly (Figure 14.11(b)). In contrast, 4C (cf. Figure 14.11(c), parameters optimized to $MinPts = 20$, $\varepsilon = 0.1$, $\lambda = 2$, and $\delta = 0.01$) and ORCLUS (cf. Figure 14.11(d), parameters $k = 5$ and $l = 2$ reflect the cluster structure exactly) could not compete. Both missed very large and important parts of the clustering structure. This bad behavior of the two competitors can partly be explained by the high degree of noise present in the data set. The influence of noise on the existing approaches can be observed in Figure 14.12. Omitting the noise points, 4C is able to detect the cluster structure relatively well (cf. Figure 14.12(a)) but cannot handle intersecting clusters. Even on the data set without noise points, ORCLUS was not able to identify the 5 clusters correctly (cf. Figure 14.12(b)). This again illustrates the superiority of CASH over existing methods especially in terms of noise robustness.

14.10.3 Real-World Data

We applied CASH on the Wages data set[1], a data set containing average career statistics of current and former NBA players[2] and a gene expression data set [127]. The Wages data consist of 534 4D observations (A=age, YE=years of education, YW=years of work experience, and W=wage) from the 1985 Current Population Survey. As parameters for CASH we used $\mu = 70$ and $s = 40$. The results are summarized in Table 14.1: CASH detected three pure subspace clusters in this data set, two data objects have

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

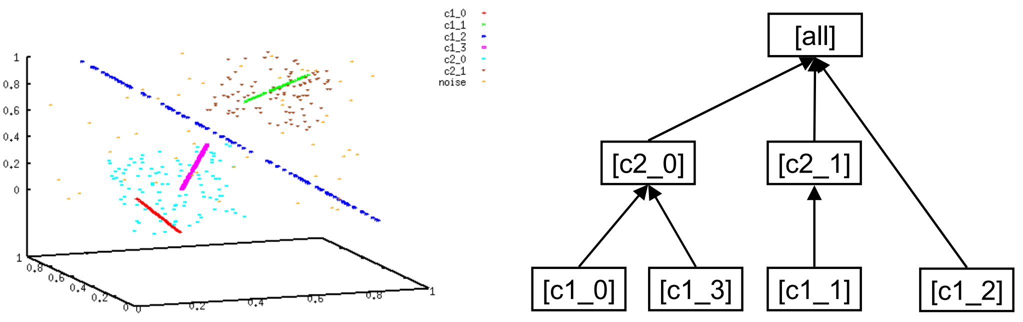
[2]obtained from `http://www.nba.com`

Table 14.2: CASH clustering on NBA data.

c ID	dim	Description
1	1	“go-to-guys”
2	2	shooting guards
3	2	point guards
4	2	starting centers
5	8	point guards
6	9	power forwards
7	9	small forwards
8	10	well-known rebounder
9	12	role players/reserves

been identified as noise objects. The first cluster consists only of people having 12 years of education and having started their working life at the age of 18. The second cluster consists only of people having 16 years of education and having started their working life at the age of 22. 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.

The NBA data contains 15 statistical measures such as “games played” (G), “games started” (GS), “minutes played per game” (MPG), “points per game” (PPG), etc. for 413 former and current NBA players. As parameters for CASH we used $\mu = 30$ and $s = 45$. CASH detected 9 interesting clusters of very different dimensionality each containing players of similar characteristics (cf. Table 14.2). In addition, several players were noise. The detected correlations confirmed basketball fundamentals. For example, in cluster 1 containing superstars like Michael Jordan, Larry Bird, Shaquile O’Neal, and James Worthy, PPG of all players were negatively dependent on G and MPG. On the other hand, the more games the players were in (G), the higher the number of starting line-up appearances (GS). Let us note that this cluster also contains less well-known players that had similar characteristics such as Rik Smits, Dan Majerle, and Rick Fox. The three clusters containing guards all showed correlations between G and MPG on the one hand, and



(a) Data set DS3. (b) Hierarchy detected by CASH.

Figure 14.13: Hierarchies found on synthetic data set DS3.

the number of assists and steals per game on the other hand. For the guards in cluster 3, this correlation was positive, whereas for the guards in cluster 5, this correlation was negative. On the other hand, cluster 3 exhibits a positive correlation between the G and GS. In cluster 5 these two attributes are also correlated but in a negative fashion. This indicates that the coaches in the NBA usually decided to start with the better point guards.

In the gene expression data set (24 dimensions, 4,000 genes) CASH found several clusters of functionally related genes that are biologically interesting and relevant according to three biologically proven criteria including (i) known direct interactions of the genes or the according gene products, (ii) known common complexes of the genes or the according gene products, and (iii) participation of the according gene products in common pathways.

Neither ORCLUS nor 4C were able to detect meaningful clusters in our real-world data sets. One reason for this may be that the found clusters are highly overlapping. Thus, neither ORCLUS nor 4C can learn the appropriate similarity measure capturing the subspaces of the clusters from the local neighborhood.

14.10.4 Alternative Parametrization and Cluster Hierarchies

Last but not least, we investigated the possibilities of our novel method to produce a hierarchy of correlation clusters. We applied CASH on a 3-dimensional data set "DS3" shown in Figure 14.13(a). The data set contains several correlation clusters including four 1-dimensional clusters, a 2-dimensional cluster with two embedded 1-dimensional clusters and a 2-dimensional cluster with one embedded 1-dimensional cluster. Some noise points are also added. All clusters do not exhibit a perfect correlation, i.e. the points of the cluster deviate from the common cluster hyperplane by a small degree. In this experiment we used the alternative parametrization as described in Section 14.9 where the allowed deviation of the hyperplane can be specified independently from the allowed variance of the orientation of the cluster hyperplane. Parameters were $s = 8$, $\delta\_jitter = 0.0011$ (specifying the allowed deviation from the cluster hyperplane), and $\mu = 90$. With this alternative parametrization, CASH had no problems to recognize the true cluster structure. In contrast, using the original parametrization, we could not find a parameter setting for which CASH achieved 100% accuracy. In summary, all our experiments indicate that generally, the alternative parametrization achieves at least the same accuracy compared to the original parametrization and – in some cases – is even superior.

In addition, the correct relationships between all correlation clusters have been detected. The resulting hierarchy among the clusters reported by CASH is displayed in Figure 14.13(b). The root (level 0) of the hierarchy represents a 3-dimensional "dummy cluster" containing the entire database denoted by "all". All other clusters are obviously contained in this "cluster". On level 1, we have the two 2-dimensional clusters "c2_0" and "c2_1". On level 2 we have the four 1-dimensional clusters. The edges indicate that clusters "c1_0" and "c1_3" are contained in cluster "c2_0", cluster "c1_1" is contained in cluster "c2_1", and cluster "c1_2" is not contained in any higher dimensional cluster (except the root).

Part V

A Quantitative Model for Correlation Clusters

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.

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. It corresponds to the marriage of two widespread ideas: First, correlation analysis performed e.g. by principle component analysis (PCA) and, second, clustering which aims at identifying local subgroups of data objects sharing high similarity. Correlation clustering groups the data set into subsets called correlation clusters such that the objects in the same correlation cluster are all associated to a common hyperplane of arbitrary dimensionality. In addition, many algorithms for correlation cluster analysis also require the objects of a cluster to exhibit a certain density, i.e. feature similarity.

Correlation clustering has been successfully applied to several application domains (see e.g. [11, 144, 35]). For example, costumer 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 correlation 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 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 correlation cluster analysis, this second step is urgently needed. All existing approaches to correlation clustering usually focus only on the first step of detecting the clusters. To the best of our knowledge, there is no method for the second step of extracting quantitative correlation cluster information.

In this part, we describe an approach to handle this second step of data analysis. We introduce general concepts for extracting quantitative information on the linear dependencies within a correlation cluster such that domain experts are able to understand the correlations and dependencies in their data. In fact, our method can be applied to any correlation clusters, regardless of what correlation clustering algorithm produced the results. As output, we obtain a set of linear equations 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, we additionally introduce a framework to predict the probability that a new object is generated by a specific model of the derived ones.

This part is organized as follows. In Chapter 15 we review existing approaches for deriving descriptions of quantitative dependencies among several attributes. Our concepts to derive quantitative models of correlation clusters are proposed in Chapter 16. Chapters 17 and 18 discuss possible applications of a model for correlation clusters on classification and outlier detection, respectively.

Chapter 15

Related Work

Let us note that none of the approaches to correlation clustering surveyed so far provides a cluster model including an explicit description of the correlations within the cluster. However, there are two areas remotely related to the ideas described in this part which we will shortly sketch in this chapter.

15.1 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. [128]) 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. [139]). Discretization of attributes, moreover, does not overcome the restriction to axis parallel dependencies. Recently, Rückert et al. [118] 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 [56]. However, these association rules do not yet uncover continuous linear dependencies, but stick to certain thresholds, reflecting the boundaries of half-spaces. Thus, these approaches appear to be more related to certain types of biclustering (see Section 5.1.4).

15.2 Regression Analysis

A task very similar to the one tackled in this part is linear and multiple regression analysis (e.g. cf. [59] for an overview). 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. 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 our 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, we want to identify these dependencies when deriving a quantitative model for each cluster. Obviously, we cannot define any independent variable(s), i.e. we cannot derive a set of predictor variables. Thus, regression analysis cannot be applied to derive quantitative models for correlation clusters as envisioned in this part.

Chapter 16

Deriving Quantitative Models for Correlation Clusters

In this chapter, first a formalization of correlation clusters is presented (similar to previous formalizations) in Section 16.1, suitable to base the concepts for deriving quantitative models as described in Section 16.2. Some considerations guiding the interpretation of the derived models are presented in Section 16.3. Finally, the concepts presented in this chapter are evaluated in Section 16.4.

The material presented in this and the subsequent chapter has been published in [4].

16.1 Formalization of Correlation Clusters

In the following we assume $\mathcal{D}$ 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 cluster is a subset of those feature vectors exhibiting certain properties, e.g. the members of a cluster may be close to each other in the feature space compared to non-members, or – in case of correlation clustering – they may be close to a common regression line, while other points are not. Generally, clustering algorithms as those discussed above (Part III) can provide (implicitly or explicitly) a description of the found clusters by means of a *covariance matrix* per cluster.

Formally, let $\mathcal{C}$ be a cluster, i.e. $\mathcal{C} \subseteq \mathcal{D}$, and $\bar{x}_{\mathcal{C}}$ denote the centroid (mean) of all points $x \in \mathcal{C}$. The *covariance matrix* $\boldsymbol{\Sigma}_{\mathcal{C}}$ of $\mathcal{C}$ is defined as:

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

In general, the covariance matrix describes a distribution of attributes. EM-like algorithms utilize such a description of a distribution of attributes to derive a Gaussian model that may have created the observed data. In case of correlation clusters, however, a far more adequate description may be possible. Indeed, the fact, that correlations between features have been found, even disqualifies the covariance matrix as an adequate model of a correlation cluster, since it is sort of a probabilistic model of scatter around a certain mean value. Strong correlations as in correlation clusters, on the other hand, do suggest not only probabilistic scatter, but linear dependencies, and (by a higher perspective of interpretation) perhaps even functional or causal relations. Thus, we will now consider the intrinsic properties of correlation clusters, and how to make use of them in order to derive a more appropriate model covering dependencies quantitatively.

Consider a correlation cluster $\mathcal{C}$ that is derived using any algorithm capable of finding correlation clusters. Since the covariance matrix $\boldsymbol{\Sigma}_{\mathcal{C}}$ of $\mathcal{C}$ is a square matrix, it can be decomposed into the *eigenvalue matrix* $\boldsymbol{E}_{\mathcal{C}}$ of $\boldsymbol{\Sigma}_{\mathcal{C}}$ and the *eigenvector matrix* $\boldsymbol{V}_{\mathcal{C}}$ of $\boldsymbol{\Sigma}_{\mathcal{C}}$ such that

$$\boldsymbol{\Sigma}_{\mathcal{C}} = \boldsymbol{V}_{\mathcal{C}} \cdot \boldsymbol{E}_{\mathcal{C}} \cdot \boldsymbol{V}_{\mathcal{C}}^{\mathsf{T}}$$

The eigenvalue matrix $\boldsymbol{E}_{\mathcal{C}}$ is a diagonal matrix holding the eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{C}}$ in decreasing order in its diagonal elements. The eigenvector matrix $\boldsymbol{V}_{\mathcal{C}}$ is an orthonormal matrix with the corresponding eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{C}}$.

Now we define the correlation dimensionality of $\mathcal{C}$ as the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes in $\boldsymbol{V}_{\mathcal{C}}$ (based on the intuitions presented in Chapter 9). 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. 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 total variance of the points in $\mathcal{C}$, i.e., 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 $\mathcal{C}$:

Definition 16.1 (correlation dimensionality)
Let $\alpha \in]0,1[$. Then the correlation dimensionality $\lambda_{\mathcal{C}}$ *of a set of points $\mathcal{C}$ is the smallest number r of eigenvalues e_i in the $d \times d$ eigenvalue matrix $\boldsymbol{E}_{\mathcal{C}}$ explaining a portion of at least α of the total variance:*

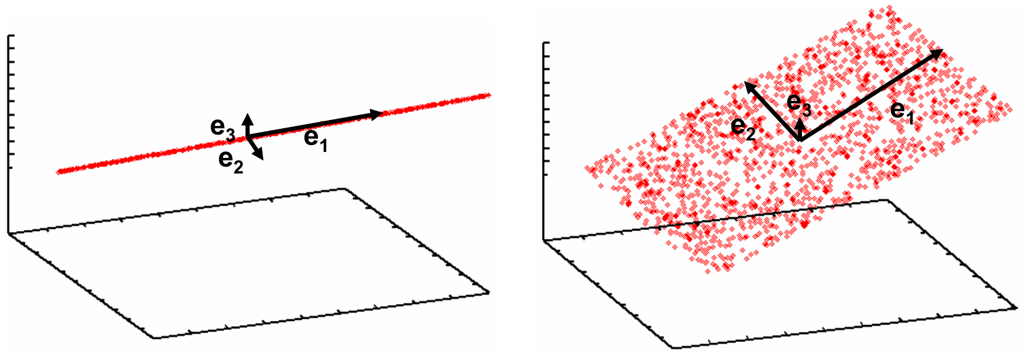
$$\lambda_{\mathcal{C}} = \min_{r \in \{1,\ldots,d\}} \left\{ r \;\middle|\; \frac{\sum_{i=1}^{r} e_i}{\sum_{i=1}^{d} e_i} \geq \alpha \right\}$$

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 set $\mathcal{C}$ allowing a small deviation corresponding to the remaining portion of variance of $1 - \alpha$. The remaining, neglected variance scatters along the eigenvectors $e_{\lambda_{\mathcal{C}}+1}, \ldots, e_d$.

We therefore distinguish between two disjoint sets of eigenvectors:

Definition 16.2 (strong and weak eigenvectors)
We call the first $\lambda_{\mathcal{C}}$ eigenvectors of $\boldsymbol{V}_{\mathcal{C}}$ strong eigenvectors. *The strong*



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

Figure 16.1: Correlation dimensionality of correlation clusters.

eigenvectors of $\boldsymbol{V}_{\mathcal{C}}$ are denoted by $\check{\boldsymbol{V}}_{\mathcal{C}}$. The remaining eigenvectors are called weak eigenvectors*. We denote the weak eigenvectors by $\hat{\boldsymbol{V}}_{\mathcal{C}}$.*

For an illustration see Figure 16.1: in the correlation cluster of correlation dimensionality 1 (Figure 16.1(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 16.1(b)) e_1 and e_2 are *strong eigenvectors* whereas e_3 is a *weak eigenvector*. The eigenvectors are overexemplified in this example. Suppose they were scaled by their corresponding eigenvalues. If no variance remains along an eigenvector, as it may e.g. appear for e_2 and e_3 in Figure 16.1(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 to directly inspect the underlying dependencies among attributes numerically, as we will show in more detail subsequently.

16.2 Deriving Quantitative Models for Correlation Clusters

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) $\bar{x}_{\mathcal{C}} = (\bar{x}_1 \cdots \bar{x}_d)^{\mathsf{T}}$ of all points belonging to cluster $\mathcal{C}$ and the set of weak eigenvectors, $\hat{\boldsymbol{V}}_{\boldsymbol{\mathcal{C}}}$, that are normal vectors to the hyperplane. Then we can derive the following equation system to describe the hyperplane, consisting of $d - \lambda$ equations:

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

where $v_{i,j}$ is the value at column i, row j in the eigenvector matrix $\boldsymbol{V}_{\boldsymbol{\mathcal{C}}}$ of $\mathcal{C}$. As we have pointed out, only the weak eigenvectors are relevant. Thus we can equivalently denote this equation system by

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

The defect of $\hat{\boldsymbol{V}}_{\boldsymbol{\mathcal{C}}}^{\mathsf{T}}$ 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 [149]. 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, we suppose to use total pivoting for the Gauss-Jordan elimination.

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 using retained data points. Besides, as we will see in the next chapter, it may also serve as a predictive model to classify new data points.

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

1. Run a clustering algorithm on $\mathcal{D}$ that is able to find correlation clusters, i.e. use e.g. 4C or ORCLUS. However, also k-means or DBSCAN is possible, provided that a proper distance function taking into account the correlation dimension is used. If the result may be restricted to clusters of *positively* correlated features, even the usage of any general biclustering or pattern-based clustering algorithm 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 our experiments we use COPAC [6], as it has been shown to improve over 4C as well as ORCLUS w.r.t. efficiency, effectiveness, and robustness.

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

   (a) Derive the covariance matrix $\boldsymbol{\Sigma}_{\mathcal{C}_i}$.

   (b) Select the weak eigenvectors $\hat{\boldsymbol{V}}_{\mathcal{C}_i}$ of $\boldsymbol{\Sigma}_{\mathcal{C}_i}$ with respect to a certain α.

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

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

   (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.

16.3 Interpretation of Correlation Cluster Models

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

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

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 we have linear dependencies 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 we must not draw 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 we have to rely on the domain knowledge of experts. However, we believe that uncovered quantitative relationships will lead to refined experiments and help to finally explore supposable causalities. Thus, we could choose experimental settings 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 $S \subset \{x_1, x_3, x_5\}$ 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 (as shown e.g. for genetic regulatory interaction networks, cf. [72]).

16.4 Evaluation

In our experiments we use the correlation clustering algorithm COPAC [6] to generate the correlation clusters in a preprocessing step to our method. We chose this algorithm due to its efficiency, effectiveness, and robustness. In each case, parameters for clustering were chosen according to the recommendations in [6]. Let us again note that any other (correlation) clustering algorithm is applicable for preprocessing.

16.4.1 Synthetic data sets

For our experiments we used several synthetic data sets 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 a

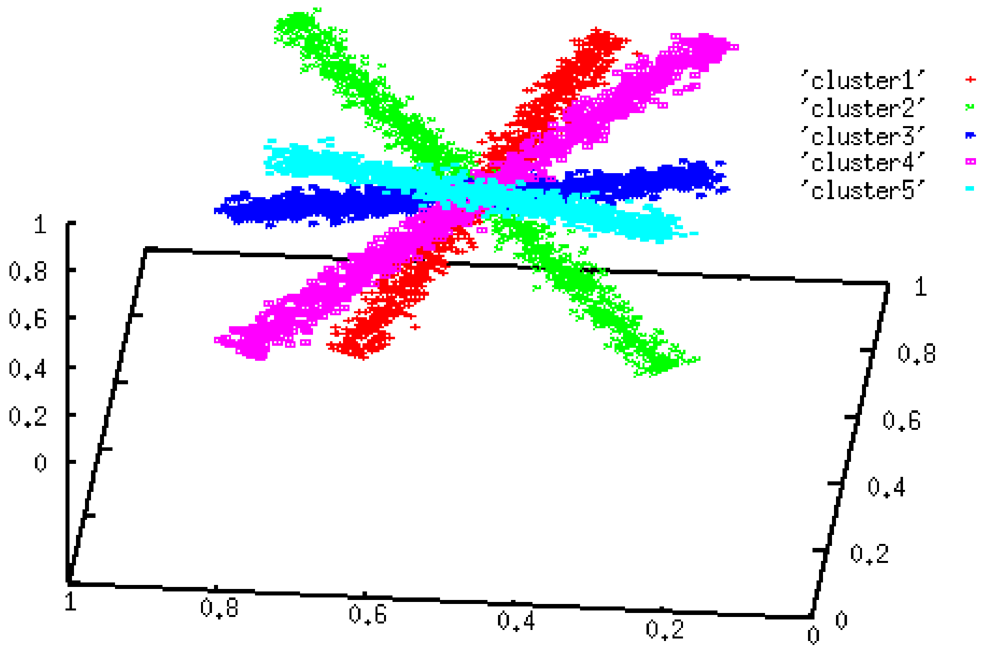
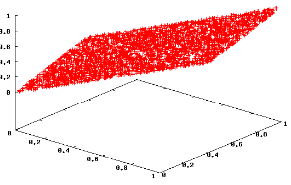


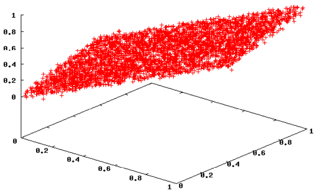
Figure 16.2: Synthetic data set DS1.

Table 16.1: Dependencies on DS1 data.

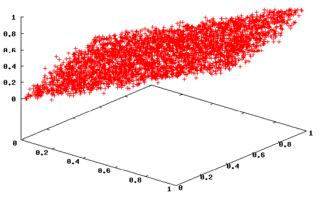
	Generated		Found
	dependencies	standard deviation	dependencies
cluster 1	$x1 - x3 = 0$ $x2 + 0.5x3 = 0.75$	$\sigma = 0.0246$	$x1 - 1.0069x3 = -0.0035$ $x2 + 0.5065x3 = 0.7537$
cluster 2	$x1 - x3 = 0$ $x2 - x3 = 0$	$\sigma = 0.0243$	$x1 - 1.0027x3 = -0.0028$ $x2 - 0.9901x3 = 0.0022$
cluster 3	$x1 + x3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



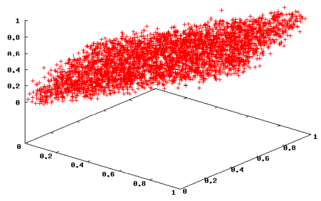
(a) DS2_0 ($\sigma_0 = 0$)



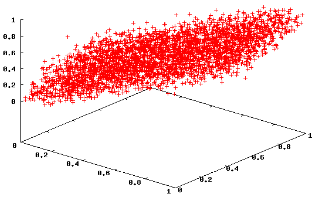
(b) DS2_1 ($\sigma_1 = 0.0173$)



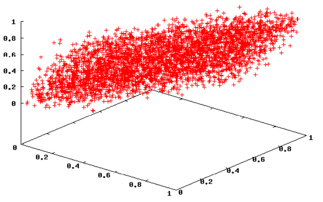
(c) DS2_2 ($\sigma_2 = 0.0346$)



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



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



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

Figure 16.3: Synthetic data sets with different values for standard deviation.

line of 1,000 points in $\mathbb{R}^3$ (cf. Figure 16.2). 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 our proposed method to obtain a quantitative model for the correlation clusters. As it can be seen in Table 16.1 we derived a good approximation of the equation systems that define the models for the correlation clusters despite the obviously strong jitter in the data set.

In the second experiment we evaluated our method on data sets with varying standard deviation. We generated six data sets (DS2_0, …, DS2_5) forming a 2-dimensional hyperplane in $\mathbb{R}^3$ with different values for the standard deviation of the distances. 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 16.3). 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

Table 16.2: Dependencies on DS2 data.

	Generated		Found
	dependencies	standard deviation	dependencies
$DS2_0$	$x1 - 0.5x2 - 0.5x3 = 0$	$\sigma = 0$	$x1 - 0.5000x2 - 0.5000x3 = 0.0000$
$DS2_1$	$x1 - 0.5x2 - 0.5x3 = 0$	$\sigma = 0.0173$	$x1 - 0.4989x2 - 0.5002x3 = 0.0000$
$DS2_2$	$x1 - 0.5x2 - 0.5x3 = 0$	$\sigma = 0.0346$	$x1 - 0.5017x2 - 0.4951x3 = 0.0016$
$DS2_3$	$x1 - 0.5x2 - 0.5x3 = 0$	$\sigma = 0.0520$	$x1 - 0.5030x2 - 0.5047x3 = -0.0059$
$DS2_4$	$x1 - 0.5x2 - 0.5x3 = 0$	$\sigma = 0.0693$	$x1 - 0,4962x2 - 0.5106x3 = -0.0040$
$DS2_5$	$x1 - 0.5x2 - 0.5x3 = 0$	$\sigma = 0.0866$	$x1 - 0.4980x2 - 0.4956x3 = 0.0064$

rather clear even if the points stronger deviate from the optimal hyperplane. In general, our proposed method has proven to be rather robust w.r.t. small jitter.

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

16.4.2 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

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

Table 16.3: Dependencies on Wages data.

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

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, we found a dependency between wage and age: 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 4610 genes at five different time slots (5, 10, 15, 30, 60 minutes) after initiating the apoptosis pathway.

We analyzed two correlation clusters detected by COPAC. The derived

[2]The confidential data set is donated by our project partners.

Table 16.4: Dependencies on Gene expression data.

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

dependencies of these clusters are depicted in Table 16.4. The attributes are abbreviated by Mi, where i 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. We also applied our method 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`

("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 are depicted in Table 16.5. Let us note 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, our 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 among the measurements.

Table 16.5: Dependencies on Wisconsin breast cancer data.

cID	derived dependencies
1	$A1 = 2, \quad A2 = 1, \quad A3 = 1, \quad A4 = 1, \quad A6 = 1, \quad A5 - 0.1 \cdot A7 = 1.9$
	$A8 = 1,$ 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	$A1 - 5.4 \cdot A5 + 1.6 \cdot A6 - 0.1 \cdot A7 + 1 \cdot A8 - 16.3 \cdot A9 = -21.1$
	$A2 + 1.7 \cdot A5 - 0.6 \cdot A6 + 0.2 \cdot A7 - 0.7 \cdot A8 - 9.9 \cdot A9 = -6.5$
	$A3 - 1.8 \cdot A5 - 0.8 \cdot A6 - 0.3 \cdot A7 - 0.7 \cdot A8 - 11.9 \cdot A9 = -8.5$
	$A4 - 2.3 \cdot A5 - 0.2 \cdot A6 + 0.2 \cdot A7 + 0.4 \cdot A8 + 8.6 \cdot A9 = 6.5$

Chapter 17

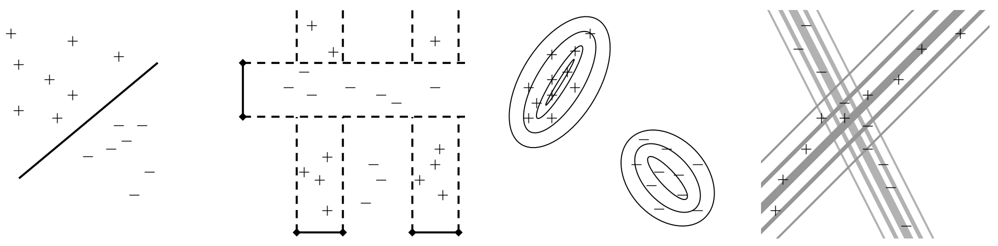
Application 1: Classification

Classification is a well established data mining task using a broad variety of techniques. However, using correlation cluster models as a basis for classification is a fundamentally new approach to classification. In this chapter, we first shortly describe the adaptation of correlation cluster models to serve as predictive models and discuss the difference to classical classification approaches (Section 17.1), and second, we evaluate the proposed concept in comparison to established classification approaches (Section 17.2).

17.1 A Classifier Based on Models for Correlation Clusters

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 to be 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 17.1: 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, that is a subspace translated by $\bar{x}_{\mathcal{C}}$, the mean vector of all points of the cluster corresponding to $\mathcal{C}$. $\text{proj}_{\mathcal{S}} : \mathbb{R}^n \to \mathbb{R}^n$ denotes the perpendicular projection of a vector to an arbitrary subspace $\mathcal{S}$ of $\mathbb{R}^n$. If $\mathcal{S}$ is given by an orthonormal basis, e.g. the set of strong eigenvectors derived for the corresponding correlation cluster, $\{s_1, \cdots, s_{\lambda_\mathcal{S}}\}$, then

$$\text{proj}_{\mathcal{S}}(x) = \langle x, s_1 \rangle s_1 + \langle x, s_2 \rangle s_2 + \cdots + \langle x, s_{\lambda_\mathcal{S}} \rangle s_{\lambda_\mathcal{S}}.$$

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, 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, our predictive models do not only provide a separating boundary between classes (cf. Figure 17.1(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 17.1(b)). The models provided by the EM algorithm or other Bayesian learners differ from our 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 17.1(c)). For underlying linear dependencies, a quadratic distance function will resemble our 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 17.1(d)).

17.2 Evaluation

As sketched above, the quantitative models generated by our method can e.g. be used to predict the class of a new object. To evaluate this potential, we used three 2-dimensional synthetic data sets each with 5 classes. 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 17.2. Note that these data sets are rather artificial and are only applied for a proof of principle.

We compared the classification accuracy of our sketched classifier to several other standard learning approaches. For this comparison we used the WEKA framework [141] with standard parameter settings, in particular, 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 17.1. As it can be

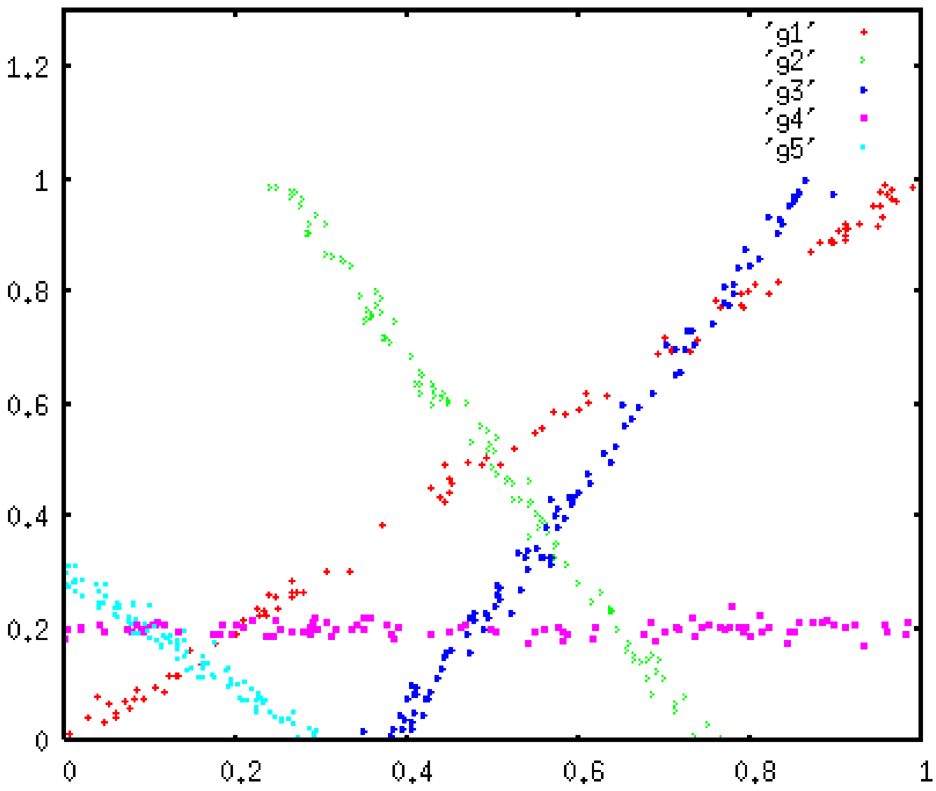


Figure 17.2: Data set $DS3_2$.

Table 17.1: Comparison of different classifiers in terms of accuracy (in %).

	Our 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

seen, our approach significantly outperforms most of the other approaches, except kNN, in terms of accuracy. Note the good results of kNN are traded for by abstaining from learning a model.

Let us 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 our concepts of linear dependencies. However, our small example may show that if the classes are generated by a model of linear dependencies as captured by our proposed concepts, our method obviously yields a better prediction accuracy than standard supervised learners.

Chapter 18

Application 2: Outlier Detection

Outlier detection is a major data mining task which aims at finding the "different mechanism" in the data. Finding outliers that do not fit well to the general data distribution is very important in many practical applications including e.g. the detection of credit card abuse in financial transactions data, the identification of measurement errors in scientific data, or the recognition of exceptional protagonists in athletic statistics.

Hawkins specifies an outlier is "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [68]. Inspired by this definition, several outlier detection schemata have been proposed over decades and research is actively ongoing. Proposed outlier detection models differ widely in how an outlier is modeled and detected (based on statistical considerations, depth contours, deviations, distances, or local divergences in different properties such as density or resolution) and in addressing the problem globally or locally. However, all those approaches somehow rely on the full dimensional Euclidean data space in order to examine the properties of each data object to detect outliers.

Usually, a real world data set contains several groups of observations (i.e. data objects) that have been generated by different mechanisms or statistical processes. These different mechanisms may show their effects in varying subsets of attributes, i.e. varying subsets of features are correlated differently for different subsets of data objects. This way, each mechanism defines a correlation of features for a local subset of data objects which we call *local correlation* throughout the rest of the paper. The mechanisms can therefore only be identified properly in various arbitrarily oriented subspaces of the original feature space rather than in the full dimensional data space.

A mechanism is supposed to have generated a minimum number of data objects in order to be considered as significant. Outliers are those objects that have not been generated by these mechanisms or processes, i.e. those objects that do not fit into the corresponding local correlations. The subset of points that show a local correlation are located on a common λ-dimensional hyperplane, where $\lambda < d$ and d is the dimensionality of the full dimensional data space. As a consequence, outliers are those objects that are not located on those hyperplanes.

This general idea is visualized in Figure 18.1. In the full dimensional (2D) space there is obviously no outlier because the object density is equal for all objects. However, all objects except for object o are located on a common hyperplane (cf. the 1D line L), i.e. these points have been generated by a common mechanism that shows its effect in a special combination of the attributes x_1 and x_2. On the other hand, object o is an outlier because o is not located on that hyperplane and, thus, is not generated by the mechanism that generates the points on the line L. We can find o as an outlier if we project the objects on the subspace S perpendicular to the line L. In that subspace, object o deviates considerably from the line L. Existing outlier detection approaches cannot identify o as an outlier because these approaches do not take any local correlations into account. In order to detect o as an outlier in the subspace S, we need a novel subspace outlier model.

Let us note that the situation in Figure 18.1 is rather idealized. In practice, we cannot expect the points that have been generated by a common mechanism to follow the corresponding correlation that strictly. In other words, the points will most likely not fit to the common hyperplane perfectly

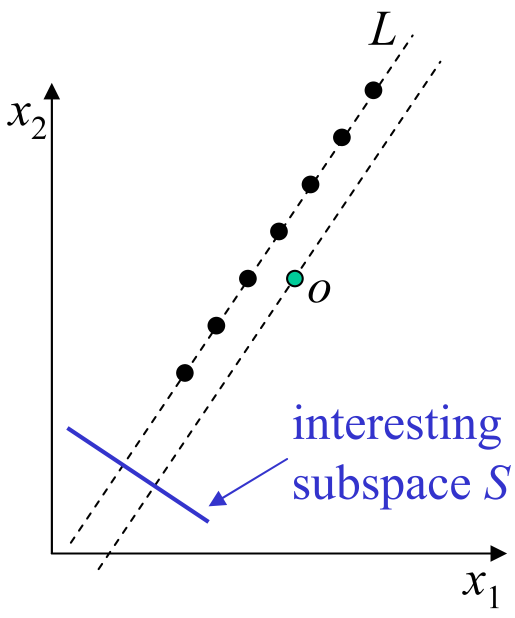


Figure 18.1: The general idea of finding outliers in subspaces of the original feature space.

but may exhibit a certain degree of jitter. This jitter has to be considered, when searching for points that deviate considerably from the common hyperplane.

In this paper, we introduce an outlier model that detects outliers as points that do not fit to any significant local correlation in the data. In contrast to existing methods, this model is the first approach to consider local correlations within the outlier detection process, i.e. to identify outliers in arbitrarily oriented subspaces of the original feature space.

The remainder of this chapter is organized as follows. We review related work in section 18.1. Our novel outlier model to detect outliers in arbitrarily oriented subspaces of the original feature space is described in Section 18.2. An experimental evaluation of the accuracy and scalability of the proposed methods in comparison to existing methods is presented in Section 18.3.

18.1 Related Work

Existing approaches for outlier detection can be classified as global or local outlier models. A global outlier approach is based on differences of properties compared over the complete data set and usually models outlierness as a binary property: for each object it is decided whether it is an outlier or not. A local outlier approach rather considers a selection of the data set and usually computes a degree of outlierness: for each object a value is computed that specifies "how much" this object is an outlier. In those applications where it is interesting to rank the outliers of the database and only retrieve the top-n outliers, a local outlier approach is obviously favorable. In such a scenario, algorithms can save computational overhead because they do not need to compute the outlierness of all objects but can prune objects that cannot be among the top-n outliers as soon as possible.

The most efficient way to tackle the problem of finding outliers is to use training data. If some objects in the database are already marked as outliers, the problem is a supervised one and any supervised learner can be used to classify unknown objects as outliers or non-outliers. An example approach for supervised outlier detection is [150]. This global approach has one obvious drawback. Usually, outlier detection is an unsupervised problem, i.e. we most often do not have any previous knowledge about the data. In the rare case where we can apply supervised techniques we further face the problem that the set of outliers is usually rather small compared to the set of non-outliers. As a consequence, the classification problem is highly unbalanced.

In statistics, outlier detection is usually addressed by a global approach that models the data by means of a multivariate Gaussian distribution and measures the Mahalanobis distance to the mean of this distribution. The classical textbook of Barnett and Lewis [22] discusses numerous tests for different distributions. Another selection of classical statistical methods for outlier detection can be found in [79]. Often, objects that have a distance of more than $3\cdot\sigma$ to the mean (σ denotes the standard deviation of the Gaussian distribution) are considered as outliers. In fact, it can be shown that the Mahalanobis distances follow a χ^2-distribution with d degrees of freedom (d

is the dimensionality of the data space). The decision rule for outlier/non-outlier can than be reformulated as follows: Objects that have a distance of more than $\chi^2(0.975)$ are considered as outliers. Though statistically sound, this method suffers from the following limitation. Usually, real-world data cannot be modeled adequately by only one Gaussian distribution. As a consequence, the mean of the distribution may be itself an outlier and judging objects by calculating their distance to the mean becomes useless. Using $k > 1$ Gaussian distributions does not solve this problem because it is usually not known beforehand how many distributions should be chosen in order to model the data adequately. In addition, since the outliers are considered when computing the mean and the standard deviation, and since both values are rather sensitive to outliers, this method is not really robust. [117] proposed a solution for the latter problem using a more robust estimation of the mean and the standard deviation.

Depth-based approaches organize data objects in convex hull layers expecting outliers from data objects with shallow depth values only [135, 119, 78]. These approaches originate from computer graphics and are infeasible for data spaces of high dimensionality due to the inherent exponential complexity of computing convex hulls.

Deviation-based outlier detection groups objects and considers those objects as outliers that deviate considerably from the general characteristics of the groups. This approach has been pursued e.g. in [17, 121]. The forming of groups at random is rather arbitrary and so are the results depending on the selected groups. Forming groups at random, however, is inevitable in order to avoid exponential complexity.

Probably the best-known definition of a global outlier is the concept of distance-based outliers introduced in [83]. An object o of a data set $\mathcal{D}$ is considered a DB(p,D)-outlier if at least a fraction p of the objects of $\mathcal{D}$ has a distance of greater than D to o. Variants of the global DB-outlier model include e.g. [115], [23], and [85]. Additionally, in [115] an algorithm for ranking DB-outliers is presented. Each object is ranked according to its distance to its k-th nearest neighbor. A partition-based algorithm is used to efficiently mine top-n outliers. An approximation solution to enable scalability with increasing data dimensionality is proposed in [15]. However, as adaptation to high dimensional data, only the time-complexity issue is tackled. The inherent problems of high dimensional data are not addressed by this or any other approach. On the contrary, the problems are even aggravated since the approximation is based on space filling curves. Another approximation based on reference points is proposed in [111]. This approximation, too, is only on low dimensional data shown to be valuable.

As an extension of the distance based outlier detection, some algorithms for finding an explanation for the outlierness of a point are proposed in [84]. The idea is to navigate through the lattice of combinations of attributes and to find the most significant combination of attributes where the point is an outlier. This is an interesting feature because an explicit and concise explanation why a certain point is considered to be an outlier (so that a user could conveniently gain some insights in the nature of the data) has not been provided by any other outlier detection model so far.

The notion of local outlierness has been introduced in [33] overcoming the limitations of a global view on outlierness. The authors introduce the density-based local outlier factor (LOF) that assigns a degree of outlierness to each object of the database. The LOF compares the density of each object o of a data set $\mathcal{D}$ with the density of the k-nearest neighbors of o. A LOF value of approximately 1 indicates that the corresponding object is located within a cluster, i.e. a region of homogeneous density. The higher the difference of the density around o is compared to the density around the k-nearest neighbors of o, the higher is the LOF value that is assigned to o. Several extensions and refinements of the basic LOF model have been proposed. In [133] the authors introduce a connectivity-based outlier factor (COF), whereas a spatial local outlier measure (SLOM) is proposed in [129]. In [76] a model is described that considers both nearest neighbors and reverse nearest neighbors of an object when estimating its density. In [77] the authors use the concept of micro-clusters to efficiently mine the top-n density-based local outliers in large databases, i.e. those n objects having the highest LOF value. A similar algorithm is presented in [76] for the proposed extension of the LOF model.

Another local outlier detection schema called Local Outlier Integral (LOCI) is proposed in [107] which is based on the concept of a multi-granularity deviation factor (MDEF). The main difference between the LOF and the LOCI outlier model is that the MDEF of LOCI uses ε-neighborhoods rather than k-nearest neighbors. The authors propose an approximative algorithm that computes the LOCI values of each database object for any ε value and displays the results as a rather intuitive outlier plot. Thereby, the approach becomes much less sensitive to input parameters. The authors further introduce an exact algorithm for outlier detection using the LOCI model.

The resolution-based outlier factor (ROF) proposed in [50] is a mix of the local and global outlier detection paradigm. The outlier schema is based on the idea of resolution change. Roughly speaking, the "resolution" specifies the number of objects considered to be neighbors of the data objects and is a data driven concept, i.e., it is based on distances rather than on parameterized concepts like k-nearest neighborhood or ε-neighborhood.

An approach claimed to be suitable especially for high dimensional data is proposed in [12]. The idea resembles a grid-based subspace clustering approach where not dense but sparse grid cells are sought to report objects within sparse grid cells as outliers. Since this is exponential in the data dimensionality, an evolutionary algorithm is proposed to search heuristically for sparse cells. Besides the complexity problems, this approach relies on full dimensional Euclidean distances similar to all other approaches reported so far. As a consequence, it is not a solution for the problems typically encountered in high dimensional data in general and for the problem of local correlations among attributes in particular.

To the best of our knowledge, no outlier detection approach has been proposed so far that considers correlations of attributes for outlier detection and searches for outliers in arbitrarily oriented subspaces of high dimensional data.

Outlier detection is orthogonal to clustering where the aim is to find a natural grouping of sets of similar data objects. In fact, clustering algorithms have similar problems like outlier detection approaches in high dimensional data. Thus, a lot of specialized methods have been proposed for clustering high dimensional data. Solutions include the detection of clusters in axis-parallel subspaces of the data space, the detection of pattern-based clusters, and the detection of clusters in arbitrarily oriented subspaces of the data space (see Part II). Although outliers can usually be seen as objects that do not fit well into any cluster, subspace clustering algorithms can usually not be used for subspace outlier detection because these algorithms search for clusters and their corresponding subspace rather than outliers and their corresponding subspaces. Any object that is not assigned to a subspace cluster need not necessarily be a remarkable outlier in any of the subspaces in which the detected clusters exist.

18.2 Outlier Detection in Subspaces

18.2.1 General Idea

In general, local outlier detection models have shown better accuracy than global models. Existing techniques for local (and also global) outlier detection do not consider correlations of features. This is a severe limitation in many applications where correlations among attributes indicate different mechanisms or statistical processes in the data. For example, in many scientific applications the relationship between causation and effect can only be exploited when considering correlations among attributes.

Here, we try to overcome this limitation of existing approaches by proposing a local outlier model that considers correlations. In the following we assume $\mathcal{D}$ to be a database of n feature vectors in a d-dimensional real-valued feature space, i.e. $\mathcal{D} \subseteq \mathbb{R}^d$. Data objects that show a similar correlation among some attributes are located on a common λ-dimensional ($\lambda < d$) hyperplane, hereafter also called *correlation hyperplane*. The basic idea of our approach is that a data point o is an outlier w.r.t. a set of (local) reference objects $\mathcal{C}$ if o is not located on the hyperplane spanned by the points of $\mathcal{C}$. If the objects in $\mathcal{C}$ and o itself are projected on the (arbitrarily oriented) subspace perpendicular to the hyperplane spanned by the objects in $\mathcal{C}$, we can observe that the objects in $\mathcal{C}$ exhibit a high density in that subspace whereas o is considerably far apart from these objects. This idea is visualized in Figure 18.1. If o and the objects in $\mathcal{C}$ (located on line L) are projected onto the subspaces S perpendicular to the line that is spanned by the objects in $\mathcal{C}$, we can observe that o significantly deviates from the set of objects $\mathcal{C}$ in S. Since we implement a local approach, we will select neighbors of o as the reference set $\mathcal{C}$.

In summary, we consider a set of reference objects $\mathcal{C}$ for an object o in order to evaluate the outlier degree of o w.r.t. $\mathcal{C}$ similar to existing local outlier detection approaches. However, fundamentally different to existing approaches, we do not consider the density of o and the density of the neighbors of o in the full dimensional space. Rather, we determine the correlation, i.e. the hyperplane, defined by the neighbors of o and evaluate the deviation of o to its neighbors in the subspace perpendicular to that hyperplane. This procedure measures how good o fits to this correlation, i.e. how far apart from the hyperplane o is in the original feature space. The motivating idea for this approach is the assumption of possible dependencies among different attributes. Different mechanisms that have generated the data will then most likely exhibit also different sets of dependencies among attributes. Eventually, these linear dependencies are also interesting themselves in order to grasp possible underlying mechanisms, since those mechanisms are presumably unknown if a local outlier detection approach is performed.

In the following, we first introduce a concept to describe correlation hyperplanes (cf. 18.2.2). Based on this description, we then present our outlier model (cf. 18.2.3). We discuss the choice of the local reference set $\mathcal{C}$ such that a significant mechanism can be modeled from $\mathcal{C}$ and the weighted construction of the covariance matrix in order to derive the principal components in a robust way (cf. 18.2.4). As an innovative and highly useful concept a method to derive an explanation for the found outliers is presented (cf. 18.2.5). A description of the outlier detection algorithm based on the proposed concepts and a discussion of its properties (cf. 18.2.6) rounds up this Section.

18.2.2 Describing Correlation Hyperplanes

The basic concept of our approach is a description of a correlation hyperplane. In order to evaluate how good a point fits to the correlation hyperplane that is spanned by a reference set, we use Principal Component Analysis (PCA) as a well established concept to determine and represent any λ-dimensional hyperplane ($\lambda < d$).

For a set of points $\mathcal{C} \subseteq \mathcal{D}$ and the centroid (mean) $\bar{x}_{\mathcal{C}}$ of all points $x \in \mathcal{C}$, the *covariance matrix* $\boldsymbol{\Sigma}_{\mathcal{C}}$ of $\mathcal{C}$ is generally defined as:

$$\boldsymbol{\Sigma}_{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \cdot \sum_{x \in \mathcal{C}} (x - \bar{x}_{\mathcal{C}}) \cdot (x - \bar{x}_{\mathcal{C}})^{\mathsf{T}} \tag{18.1}$$

The covariance matrix $\boldsymbol{\Sigma}_{\mathcal{C}}$ for a set of points $\mathcal{C}$ generally describes a distribution of attributes which can be utilized to derive a Gaussian model that may have created the observed data. This way, the covariance matrix represents a probabilistic model of scatter around a certain mean value. However, in case of correlation hyperplanes a far more adequate description or model may be possible because strong correlations (as they appear in correlation hyperplanes) do not only suggest probabilistic scatter, but linear dependencies among attributes. These linear dependencies among attributes may represent (by a higher perspective of interpretation) functional or causal relations. Thus, we argue not to use the covariance matrix to model the correlation among a set of points $\mathcal{C}$. Rather, we will consider the intrinsic properties of correlation hyperplanes, and elaborate how to make use of them in order to derive a more appropriate model covering dependencies even quantitatively. Let us note that this model cannot only be used to define outliers w.r.t. correlation hyperplanes but also to derive qualitative knowledge about the outliers in the sense of specifying the correlations w.r.t. to which the objects are outliers.

The covariance matrix $\boldsymbol{\Sigma}_{\mathcal{C}}$ of any point set $\mathcal{C}$ can be decomposed into the *eigenvalue matrix* $\boldsymbol{E}_{\mathcal{C}}$ of $\boldsymbol{\Sigma}_{\mathcal{C}}$ and the *eigenvector matrix* $\boldsymbol{V}_{\mathcal{C}}$ of $\boldsymbol{\Sigma}_{\mathcal{C}}$ such that

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

The eigenvalue matrix $\boldsymbol{E}_{\mathcal{C}}$ is a diagonal matrix holding the eigenvalues of $\boldsymbol{\Sigma}_{\mathcal{C}}$

in decreasing order in its diagonal elements. The eigenvector matrix $\boldsymbol{V}_{\mathcal{C}}$ is an orthonormal matrix with the corresponding eigenvectors of $\boldsymbol{\Sigma}_{\mathcal{C}}$.

We denote the number of dimensions of the (arbitrarily oriented) subspace which is spanned by the major axes in $\boldsymbol{V}_{\mathcal{C}}$ as the *correlation dimensionality* of $\mathcal{C}$. In order to compute the correlation dimensionality of $\mathcal{C}$ we have to determine the principal components (eigenvectors) of the points in $\mathcal{C}$. The eigenvectors are sorted by descending eigenvalues, i.e. the eigenvector associated with the largest eigenvalue e_1 represents the first principal component, the eigenvector associated with the second largest eigenvalue e_2 determines the direction of the second principal component and so on. The sum of the eigenvalues e_i $(1 \leq i \leq d)$ equals the trace of the square matrix $\boldsymbol{\Sigma}_{\mathcal{C}}$ which is the total variance $VAR(\mathcal{C})$ of the points in $\mathcal{C}$, i.e.,

$$VAR(\mathcal{C}) = \sum_{i=1}^{d} e_i.$$

Thus, each obtained eigenvalue represents the fraction of the variance explained by the corresponding principal component, in decreasing order of importance. If all components would have equal importance, each eigenvalue would exactly explain $\frac{1}{d}$ of the total variance $VAR(\mathcal{C})$. In case of a correlation hyperplane, only a subset of components is important. The number of these important components equals the correlation dimensionality. Thus, the correlation dimensionality of a set of points $\mathcal{C}$ is intuitively defined as the number of eigenvectors v_i such that all these v_i explain more than $\frac{1}{d}$ of the total variance $VAR(\mathcal{C})$ each. These ideas are illustrated in Figure 18.2. Figure 18.2(a) shows a set of points $\mathcal{C}$ that span a correlation hyperplane H of correlation dimensionality 1 corresponding to a (perfect) line. One eigenvector (v_1) already explains the total variance of $\mathcal{C}$ and, thus, more than $\frac{1}{d} \cdot VAR(\mathcal{C})$. Figure 18.2(b) shows a set of points $\mathcal{C}$ that span a correlation hyperplane H of correlation dimensionality 2 corresponding to a (perfect) plane. Here, two eigenvectors explain the total variance of $\mathcal{C}$ and each of them explain more than $\frac{1}{d}$ of the total variance.

Let us 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. In real-world data sets, this is obviously

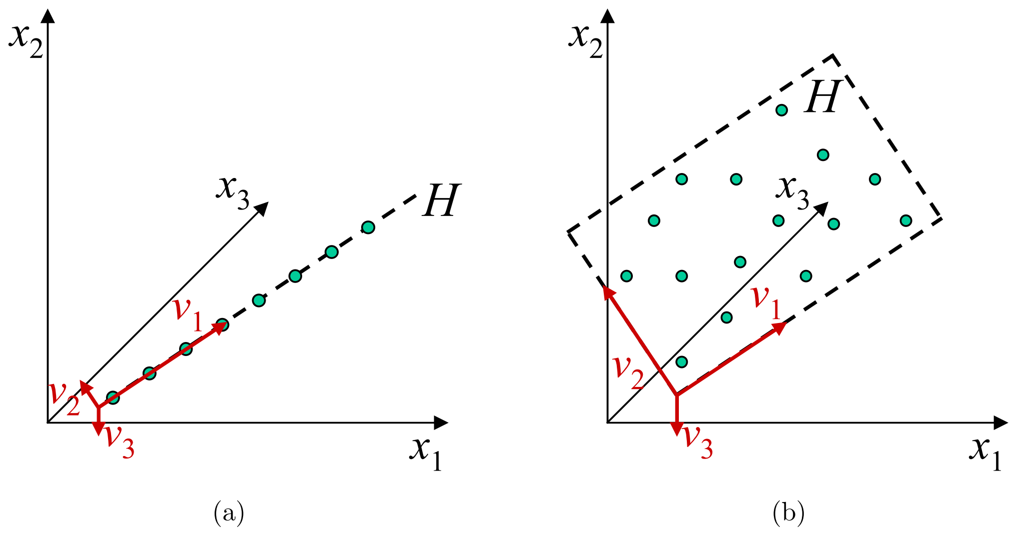


Figure 18.2: Illustration of the correlation dimensionality.

a rather unrealistic scenario. The points will most likely deviate from the (idealized) hyperplane. As a consequence, it is not suitable to select only those components that each explain the expected part of the total variance of the points (because in these cases, there are no such components). Rather, we require an important component to explain more than a certain percentage α of the expected portion of the total variance, i.e. $\alpha \cdot \frac{1}{d} \cdot VAR(\mathcal{C})$. For $\alpha = 1$, the threshold exactly reflects the expected variance covered by a single eigenvector for perfectly uniformly distributed data. This way, the correlation dimensionality represents the dimensionality of a hyperplane neglecting a certain amount of deviation in orthogonal direction.

Definition 18.1 (correlation dimensionality)
The correlation dimensionality $\lambda_{\mathcal{C}}$ *of a set of points* $\mathcal{C}$ *is the number* $r \in \{1, \ldots, d\}$ *of those eigenvectors* v_i *that each explain more than a percentage* $\alpha \in \mathbb{R}^+$ *of the expected part of the total variance:*

$$\lambda_{\mathcal{C}} = \left| \left\{ v_i \,|\, e_i > \alpha \cdot \frac{1}{d} \cdot VAR(\mathcal{C}), \quad \textit{where } 1 \leq i \leq d \right\} \right|.$$

We call the first $\lambda_{\mathcal{C}}$ *eigenvectors of* $\boldsymbol{V}_{\mathcal{C}}$*, i.e. those eigenvectors explaining more than the expected part of the total variance,* strong eigenvectors. *The* *strong eigenvectors of $\boldsymbol{V}_{\mathcal{C}}$ are denoted by $\check{\boldsymbol{V}}_{\mathcal{C}}$. The remaining eigenvectors are called* weak eigenvectors*, denoted by $\hat{\boldsymbol{V}}_{\mathcal{C}}$.*

For α, a fixed value of 1.1 showed stable results. Thus, α needs not be considered an issue for parametrization but could easily be assumed being a fixed value. Nevertheless, a higher value of α allows for an adjustment to generally noisy data. A lower value allows for an adjustment to data where a paramount axis of correlation would otherwise cover axes of interesting correlations with smaller variance. Note that this reasoning differs subtly from the assessment of the correlation dimensionality presented in previous chapters.

The $\lambda_{\mathcal{C}}$-dimensional affine subspace 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 $\bar{x}_{\mathcal{C}}$ defines the *correlation hyperplane* $H_{\mathcal{C}}$ of $\mathcal{C}$. In other words, the correlation dimensionality $\lambda_{\mathcal{C}}$ is the dimensionality of the affine subspace containing all points of the set $\mathcal{C}$ allowing a small deviation. The remaining, neglected variance scatters along the eigenvectors $v_{\lambda_{\mathcal{C}}+1}, \ldots, v_d$.

Using the concepts introduced in Chapter 16, we can describe this correlation hyperplane $H_{\mathcal{C}}$ by a unique equation system:

$$\hat{\boldsymbol{V}}_{\mathcal{C}}^{\mathsf{T}} \cdot x = \hat{\boldsymbol{V}}_{\mathcal{C}}^{\mathsf{T}} \cdot \bar{x}_{\mathcal{C}}. \tag{18.2}$$

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 points in $\mathcal{C}$ and, therefore, allows to determine whether another point deviates strong enough from this model to be supposedly generated by a different process. While the model is defined in a deterministic way, the deviation may be more adequately modeled as a stochastic process. In combination, as we will see next, these two components yield a subspace outlier model suitable both, to approximately describe a process possibly responsible for the reference set of objects for an object o, and to assign to o a degree of outlierness w.r.t. this process, i.e., a probability of being an outlier.

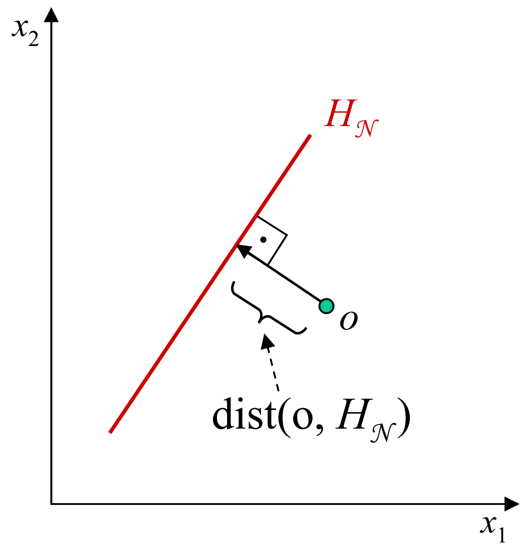


Figure 18.3: Illustration of the distance of an object o to a hyperplane $H_{\mathcal{C}}$.

18.2.3 Subspace Outlier Model

Having derived a descriptive model, it can be refined by determining an average distance of points in $\mathcal{C}$ from the correlation hyperplane $H_{\mathcal{C}}$, i.e. in the subspace perpendicular to $H_{\mathcal{C}}$. As discussed above, due to jitter, such deviations of points in $\mathcal{C}$ from $H_{\mathcal{C}}$ are typically to be expected in natural systems, e.g. due to errors in measurement. The distance of an object o to a hyperplane $H_{\mathcal{C}}$ is thereby naturally defined as the Euclidean distance to its perpendicular projection onto the hyperplane, i.e.:

$$dist(o, H_{\mathcal{C}}) = ||o - \bar{x}_{\mathcal{C}} - \pi_{H_{\mathcal{C}} - \bar{x}_{\mathcal{C}}}(o - \bar{x}_{\mathcal{C}})||,$$

where $\pi_{\mathcal{S}} : \mathbb{R}^n \to \mathbb{R}^n$ denotes the perpendicular projection of a vector onto an arbitrary subspace $\mathcal{S}$ of $\mathbb{R}^n$. If $\mathcal{S}$ is given by an orthonormal basis, e.g. the set of strong eigenvectors derived for the corresponding correlation hyperplane, $\{v_1, \cdots, v_{\lambda_{\mathcal{S}}}\}$, then

$$\pi_{\mathcal{S}}(x) = \langle x, v_1\rangle v_1 + \langle x, v_2\rangle v_2 + \cdots + \langle x, v_{\lambda_{\mathcal{S}}}\rangle v_{\lambda_{\mathcal{S}}}.$$

The idea of this distance $dist(o, H_{\mathcal{C}})$ is illustrated in Figure 18.3. Assuming the distances $dist(o, H_{\mathcal{C}})$ of all $o \in \mathcal{C}$ fit to a Gaussian distribution

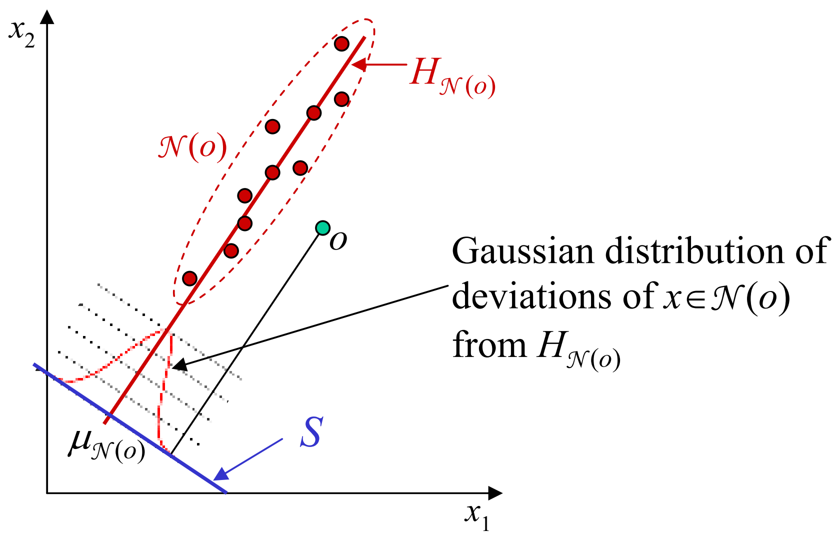

x_2
$H_{\mathcal{N}(o)}$
$\mathcal{N}(o)$
o
Gaussian distribution of deviations of $x \in \mathcal{N}(o)$ from $H_{\mathcal{N}(o)}$
$\mu_{\mathcal{N}(o)}$
S
x_1

Figure 18.4: Gaussian distribution of the distances of all points in $\mathcal{C}$ to $H_\mathcal{C}$.

with $\mu_\mathcal{C} = 0$, the standard deviation $\sigma_\mathcal{C}$ of these distances suffices to define a Gaussian model of deviations from the common correlation hyperplane (cf. Figure 18.4). The probability density function of this distribution is given by

$$f(o) = \frac{1}{\sigma_\mathcal{C}\sqrt{2\pi}} e^{-\frac{1}{2\sigma_\mathcal{C}^2} dist(o,H_\mathcal{C})^2}.$$

This probability density is a very intuitive measurement for the degree of outlierness of any $o \in \mathcal{D}$ w.r.t. the set of points in $\mathcal{C}$. However, in order to be able to interpret this probability density, one needs to scale it because the probability density depends on the standard deviation of the underlying distribution. As a consequence, a point o may strongly deviate from a corresponding hyperplane $H_\mathcal{C}$ but due to a plain Gaussian curve (i.e. a high standard deviation) the value of $f(o)$ may still be rather high. This problem is visualized in Figure 18.5. Three probability density functions f_1, f_2, and f_3 with varying standard deviations obviously return very different values $f_1(x)$, $f_2(x)$, and $f_3(x)$, respectively, for a given observation x.

A resulting value reflecting the outlier degree in the range of $[0, 1]$ would be more useful, since this could be interpreted as "outlier probability" of a given point and probabilities of different points are comparable so that e.g.

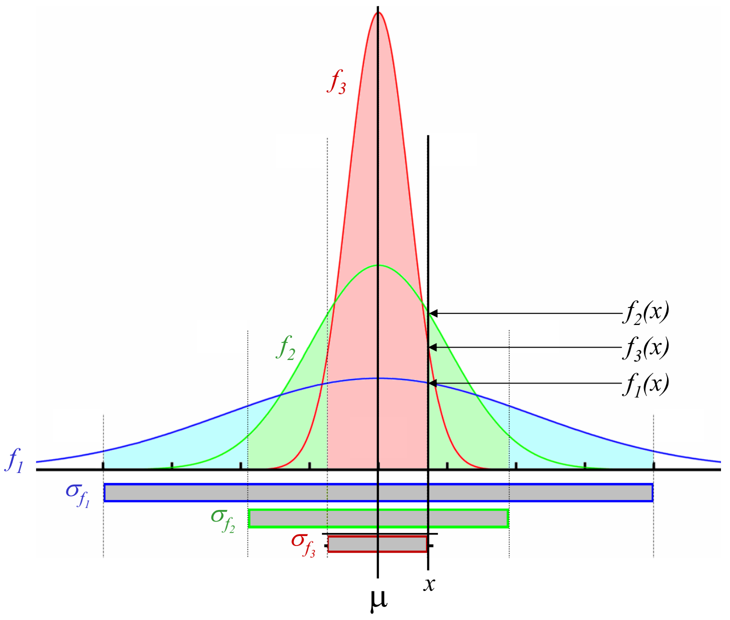


Figure 18.5: Probability density functions with varying standard deviation.

a ranking of the top outliers can be computed. Note that, in this context, we cannot achieve such a probability as easily as in a classification-context where we simply have to decide which of several processes is most likely to have generated a given observation (cf. Chapter 17). Here, contrariwise, we have only one Gaussian process and need to decide whether or not a given observation is generated by this process.

Since we are assuming that the distances of objects $o \in \mathcal{C}$ to the hyperplane $H_\mathcal{C}$ follow a Gaussian distribution, it seems appropriate to use some Gaussian density function to compute a probability that any $o \in \mathcal{C}$ was generated by the same mechanism as the other points $o' \in \mathcal{C}$. As normalization, the standard deviation $\sigma_\mathcal{C}$ of the local neighborhood can then be used.

The cumulative density function (cdf) Φ for Gaussian distributions is known to be defined as

$$\Phi_{\mu_\mathcal{C},\sigma^2_\mathcal{C}}(x) := \frac{1}{\sigma_\mathcal{C}\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(u-\mu_\mathcal{C})^2}{2\sigma^2_\mathcal{C}}} du.$$

There is a well-known variant of this formula in statistics known as the (Gaussian) *error function* $\mathit{erf}(x)$. While this function essentially is just a scaled and normalized version of Φ, it has exactly the scaling we are interested in. The error function is commonly defined as

$$erf(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

The relationship between the Gaussian Φ function and the *erf* function is as follows:

$$\Phi_{\mu_\mathcal{C},\sigma_\mathcal{C}^2}(x) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu_\mathcal{C}}{\sigma_\mathcal{C}\sqrt{2}}\right)\right).$$

The $erf(x)$ error function still needs some scaling to account for the variance. The appropriate scaling used in statistics is $erf\left(\frac{a}{\sigma_\mathcal{C}\sqrt{2}}\right)$. When using this scaling, the function computes the probability, that the error of a single point is up to a constant within error margin a. In other words, given an error margin a, it returns the probability that points are within this margin.

Now we are actually interested in the opposite, that the given point is not part of the same distribution. Obviously, this can be computed by $1 - erf\left(\frac{a}{\sigma_\mathcal{C}\sqrt{2}}\right)$. In statistics, this term is also known as the *complementary (Gaussian) error function*, denoted by $erfc\left(\frac{a}{\sigma_\mathcal{C}\sqrt{2}}\right)$. In particular, the function is monotonously decreasing with $erfc(0) = 1$ and $\lim_{x\to\infty} erfc(x) \to 0$. This makes the resulting value — the probability that the given point is an outlier — very easy to interpret. A value of 0 indicates that the particular point o perfectly fits to the hyperplane $H_\mathcal{C}$, i.e. is no outlier, whereas a considerably higher value indicates o being an outlier.

Definition 18.2 (correlation outlier probability)
Let $\mathcal{C}$ denote a local set of reference objects. The correlation outlier probability *of $o \in \mathcal{D}$ w.r.t. $\mathcal{C}$, denoted by $COP_\mathcal{C}(o)$, is defined as*

$$COP_\mathcal{C}(o) := erfc\left(\frac{dist(o, H_\mathcal{C})}{\sigma_\mathcal{C}\sqrt{2}}\right)$$

Since $dist(o, H_\mathcal{C}) \geq 0$ and $0 \leq erfc(x) \leq 1$ for $x \geq 0$ it follows that $0 \leq COP(o) \leq 1$.

In many applications, it could also be interesting to derive a binary decision whether or not an object o is an outlier rather than assigning an outlier score to o. In this scenario, we can simply follow the statistically sound and well-established notion of outliers (cf. Section 18.1) and define outliers to be points that deviate at least $3 \cdot \sigma_{\mathcal{C}}$ from the mean of a given Gaussian distribution.

Definition 18.3 (correlation outlier)
Let $\mathcal{C}(o)$ denote the reference set of any object $o \in \mathcal{D}$. The set of outliers in $\mathcal{D}$ includes all objects o that have a distance to the corresponding hyperplane $H_{\mathcal{C}(o)}$ spanned by the points in $\mathcal{C}(o)$ of more than 3 times the standard deviation $\sigma_{\mathcal{C}}$:

$$\mathit{OutlierSet} = \left\{ o \in \mathcal{D} \mid \mathit{dist}(o, H_{\mathcal{C}(o)}) > 3 \cdot \sigma_{\mathcal{C}} \right\}.$$

As a final remark, let us note that our model is valid even if no strong eigenvectors are found at all. This case is even brought forward by setting the threshold for selecting a strong eigenvector by a factor $\alpha = 1.1$ slightly higher than the expected value $\frac{1}{d} \cdot \mathit{VAR}(\mathcal{C})$. In this case, the model relaxes to the special case of a full dimensional Gaussian model, as known from classical statistical approaches, and it assigns to each point the probability of being generated by the Gaussian distribution defined for the reference set.

18.2.4 Choosing a Reference Set

So far, we have not yet discussed how to choose the reference set $\mathcal{C}$ w.r.t. which the correlation outlier probability $\mathit{COP}(o)$ of an object $o \in \mathcal{D}$ is determined. We noted above that we strive for a local outlier detection model and that $\mathcal{C}$ should be defined as the neighbors of o. In fact, we use the k-nearest neighbors of o for some input parameter k. The set of k-nearest neighbors of o is the smallest set $\mathcal{C}_k(o) \subseteq \mathcal{D} \setminus \{o\}$ that contains at least k points from $\mathcal{D} \setminus \{o\}$ and for which the following condition holds:

$$\forall x \in \mathcal{C}_k(o), \forall x' \in \mathcal{D} \setminus \mathcal{C}_k(o) : \|o - x\| < \|o - x'\|.$$

Intuitively, k determines a threshold for the minimum number of points necessary to determine a *significant* mechanism. In addition, it should be noted that k is large enough to span a λ dimensional hyperplane. Since PCA is more stable the more points it is applied to, we suggest to choose at least $k > 3 \cdot d$ following [90]. A significantly higher value is of course suitable. On the other hand, k should not be chosen too high, because then it is likely that $\mathcal{C}$ contains points that are generated by different mechanisms themselves. In that case, PCA cannot detect a meaningful correlation hyperplane.

In order to get the principal components in a more stable and robust way, we determine a weighted covariance matrix unlike the general definition in Equation 18.1. Instead, points contribute to the covariance weighted decreasingly with increasing distance from the mean. As weighting function we use again *erfc* as described in [90]. Since we assume a Gaussian error model, *erfc* appears to be the most appropriate choice.

18.2.5 Explaining and Interpreting Outliers

By now, we are able to detect objects o that do not fit well to the mechanism that has generated the k-nearest neighbors $\mathcal{C}_k(o)$ of o. Objects o that significantly deviate from the correlation hyperplane $H_{\mathcal{C}_k(o)}$ spanned by the k-nearest neighbors $\mathcal{C}_k(o)$ of o are retrieved as outliers. Obviously, it would also be interesting for the user not only to retrieve outliers but also to get an interpretation and explanation *why* objects are considered outliers.

Although a plethora of different approaches and models for outlier detection has been proposed since decades, as discussed in Section 18.1, only some of them can provide an explanation of the outlier status of a given object. In those cases, this explanation is given only implicitly. Of course, it would be very useful for any user of an outlier detection method to get an explicit explanation why a certain point is considered an outlier and what the meaning of its outlierness is. Only one approach has been proposed to derive explicit explanations for the outlier status of an object [84]. This approach is applied to distance-based outliers that are already found.

Here, we discuss the meaning of our outlier model, the possible explanations for the outlierness of a given object based on this model, and the potential utilization of such explanations.

As indicated above, we can utilize our modeling of correlation hyperplanes not only to derive an outlier score or a binary outlier decision, but also to derive a qualitative and quantitative model that explains the correlation w.r.t. which an outlier has been identified as an outlier. Unlike the argument presented in [4] for correlation clusters, here we are not interested primarily in finding the properties of the reference set itself but rather how the properties of a potential outlier differ from the properties of the reference set. However, in a first step we need to grasp the properties of the reference set in an easily interpretable way. Recall that the correlation hyperplane $H_{\mathcal{C}}$ of a set of points $\mathcal{C}$ is described by the equation system defined in Equation 18.2. The defect of $\hat{\boldsymbol{V}}_{\mathcal{C}}^{\mathsf{T}}$ represents the number of free attributes, the other attributes may be involved in linear dependencies (correlations). Similar to models for correlation clusters [4] (cf. Chapter 16), we can reveal these dependencies exhibited within the reference set, i.e. the correlations of attributes, by transforming the equation system into a reduced row echelon form of the matrix. This unique form is conveniently interpretable by unexperienced users.

To give an example, consider the following equation system describing the properties of a reference set $\mathcal{C} \subseteq \mathbb{R}^5$:

$$\begin{aligned}
1x_1 + 0x_2 + 0x_3 + \ 0x_4 \ + e_1x_5 &= f_1 \\
0x_1 + 1x_2 + 0x_3 + d_2x_4 + e_2x_5 &= f_2 \\
0x_1 + 0x_2 + 1x_3 + d_3x_4 + e_3x_5 &= f_3
\end{aligned}$$

This would provide a quantitative model describing a correlation hyperplane accommodating the reference set $\mathcal{C}$. Obviously, the dimensionality of the hyperplane is $\lambda_{\mathcal{C}} = 2$, corresponding to the number of free attributes. This way, we can assume to have linear dependencies among the attribute sets $\{x_1, x_5\}$, $\{x_2, x_4, x_5\}$, and $\{x_3, x_4, x_5\}$, specified by the factors e_1; d_2 and e_2; and d_3 and e_3, respectively. Note that this is, of course, only an assumption supported by observed correlations. But this assumption could be examined further in refined experiments. If we assume these linear dependencies describing a deterministic component of a mechanism responsible for the reference set $\mathcal{C}$ and the Gaussian model of deviations describing the stochastic component of the mechanism, we can further determine if, how far, and in which directions a given point deviates from these properties. This will eventually also allow to qualify a possible unknown mechanism responsible for the potential outlier.

The distance of a potential outlier from the hyperplane defined by the reference set can be interpreted as an error vector. This error vector describes the minimally required movement and the exact direction of this movement of the point in order to fit it perfectly to the mechanism which generated the reference set.

Possibly, the inspection of the error vector o_{err} of an object o leads the domain scientist to the conclusion that o should not be treated as an outlier (despite a large distance from the model of the reference set) but the model needs to be corrected in order to explain the object o and the reference set by the same mechanism. In this case, the error vector could be interpreted as another strong eigenvector, spanning a model-hyperplane of dimensionality $\lambda_{\mathcal{C}}+1$ together with the strong eigenvectors $\check{\boldsymbol{V}}_{\mathcal{C}}$. One could easily supplement $d-(\lambda_{\mathcal{C}}+1)$ vectors orthonormally to all vectors in $o_{err} \cup \check{\boldsymbol{V}}_{\mathcal{C}}$ in order to get a new equation system describing a new scientific model, as discussed above.

18.2.6 Algorithm

With the concepts described in the previous subsections, we are now able to evaluate outliers by considering local correlations in the data. In addition, we can derive a model for each outlier o that explains why o is considered an outlier. In particular, the correlation w.r.t. which o has been identified as an outlier is revealed quantitatively by means of an equation system.

An efficient algorithm for computing the outlier probability of all objects $o \in \mathcal{D}$ is given in Figure 18.6. An algorithm to compute the set of outliers according to Definition 18.3 can be derived in a straightforward manner. The only input parameter is k, the number of nearest neighbors that are included into $\mathcal{C}_k(o)$, which has already been discussed above.

The algorithm computes the k-nearest neighbors $\mathcal{C}_k(o)$ of each object o which requires $O(n^2)$ time using a sequential scan but can be supported by any well-established index structure reducing the runtime to $O(n \cdot \log n)$

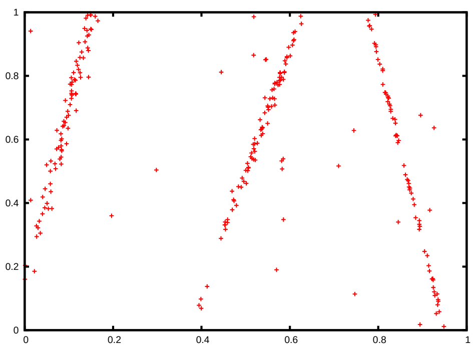
```
algorithm computeCOP
// input: the number $k$ of neighbors to determine $\mathcal{C}_k(o)$
for each $o \in \mathcal{D}$ do
    compute $\mathcal{C}_k(o)$ the $k$-nearest neighbors of $o$;
    determine $\boldsymbol{\Sigma}_{\mathcal{C}_k(o)}$ and $H_{\mathcal{C}_k(o)}$;
    determine $dist(o, H_{\mathcal{C}_k(o)})$;
    compute $COP_{\mathcal{C}_k(o)}(o)$ according to Definition 18.2;
end for
```

Figure 18.6: Computing the COP.

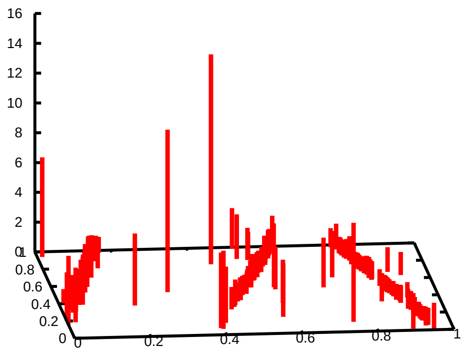
on average. In addition, for each object o the correlation hyperplane $H_{\mathcal{C}(o)}$ is determined by applying PCA and the correlation outlier probability is computed which in summary requires $O(k \cdot d^3)$ time. Since $k \ll n$, the overall runtime complexity is $O(n^2 \cdot d^3)$ without index and $O(n \log n \cdot d^3)$ when using a spatial index for nearest neighbor search. If the objects should be ranked according to decreasing COP values, this does not affect the overall runtime since sorting can be achieved in $O(n \log n)$ in the worst case. Thus, the runtime complexity is at most that of standard local outlier approaches like LOF and LOCI which both also rely on the computation of neighborhoods. However, in contrast to LOF and LOCI, our COP requires the computation of the neighbors only once for each object. LOF and LOCI compare the density of the neighborhood of each object o with the density of the neighborhood of each neighbor of o. To avoid multiple similarity queries for each object, LOF and LOCI can of course materialize the neighborhoods but this results in a huge storage overhead. In summary, COP does not have these problems and is expected to scale well even to very large databases in terms of runtime and storage cost.

18.3 Evaluation

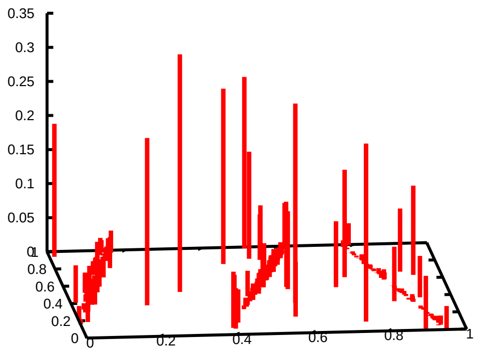
We compared COP to two existing approaches. The first competitor is LOF representing the local outlier approaches that do not consider correlations. All other existing methods in that category have similar characteristics as LOF in comparison to COP. The second approach called “Stat” is a sta-



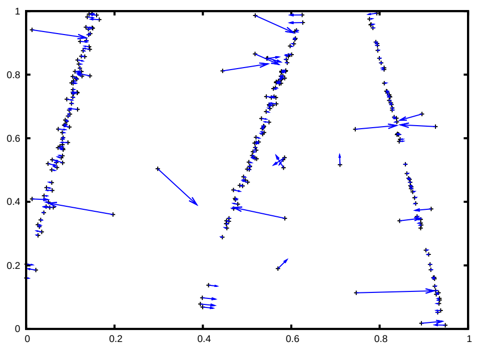
(a) Data set.



(b) Results of LOF.



(c) Results of COP.



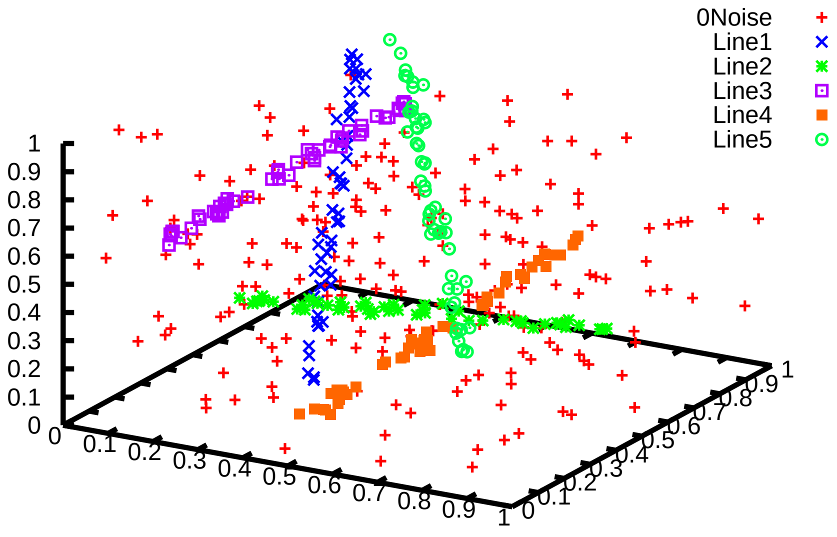
(d) Error vectors for COP.

Figure 18.7: Comparison of COP and LOF on a sample 2D data set (Both algorithms used $k = 30$).

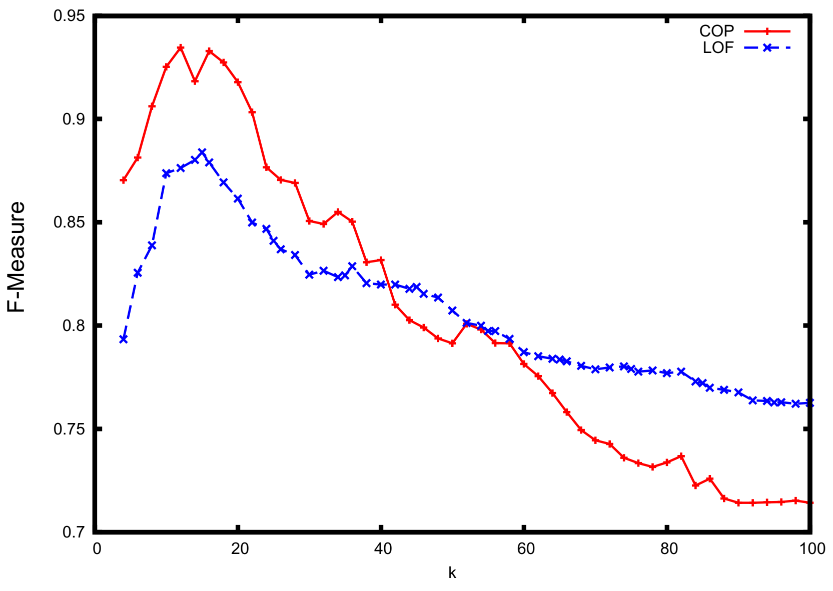
tistical approach that can be considered as a special case of COP using d weak eigenvectors for all points regardless of the correlation among the corresponding neighbors. Stat considers a point an outlier if it is sufficiently far apart from the mean of its neighbors (using the Mahalanobis distance) rather than from the correlation hyperplane spanned by the neighbors. All three competitors require only one input parameter k specifying the number of neighbors relevant for the determination of the outlier score.

18.3.1 Accuracy

First, we evaluated the differences between COP and LOF on a sample 2D toy data set (cf. Figure 18.7(a)) containing points from three different generating mechanisms located on a common 1D correlation hyperplane for each mechanism, and several outliers that deviate from these hyperplanes. The resulting LOF values are depicted in Figure 18.7(b). We can observe that LOF fails to detect outliers and non-outliers correctly in two general cases. First, if the points deviate clearly from the correlation hyperplane representing a generating mechanism but do not exhibit a lower density than the points on the hyperplane, the outliers are not detected. Second, points that are located on a hyperplane of the corresponding mechanism but are far apart from the rest of the points are mistakenly classified as outliers. In Figure 18.7(c) the COP values are plotted along the y-axis. It can be observed that our novel correlation outlier model overcomes these two limitations of LOF. Using COP, we detect only those points as outliers that deviate from the correlation hyperplane of the generating mechanism regardless of their full dimensional density. In contrast to existing techniques, the outliers found by COP can also be explained. Figure 18.7(d) illustrates for each point the "error vector", i.e. the vector that is perpendicular to the hyperplane spanned by the corresponding k-nearest neighbors. Each arrow is scaled according to the distance of the corresponding point to the corresponding correlation hyperplane. It can be seen that for most outliers, the direction of the error vector is approximately perpendicular to the hyperplane spanned by the points of the corresponding generating mechanism.

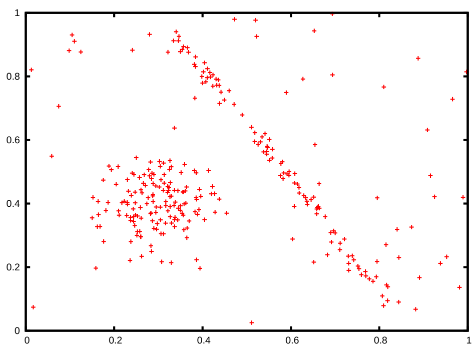


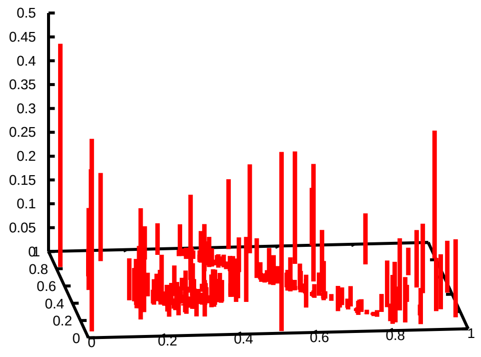
(a) Gold standard.

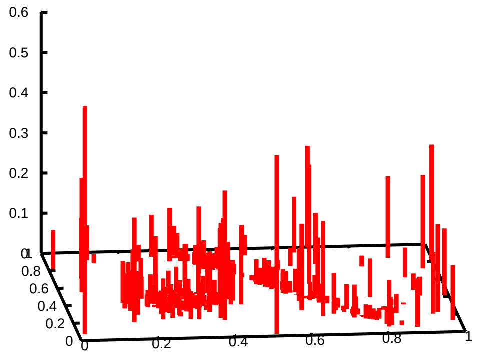


(b) F-measure of COP and LOF.

Figure 18.8: Accuracy of COP and LOF on a sample 3D data set w.r.t. parameter k.


(a) Data set.


(b) Results of Stat.


(c) Results of COP.

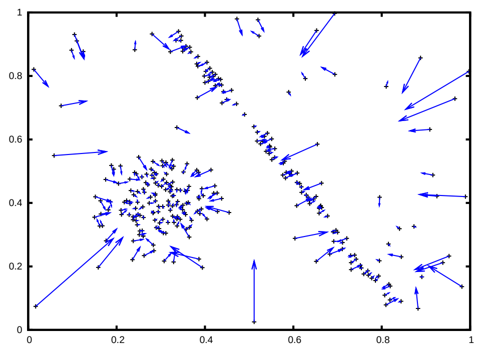

(d) Error vectors for COP.

Figure 18.9: Comparison of COP and Stat on a sample 2D data set.

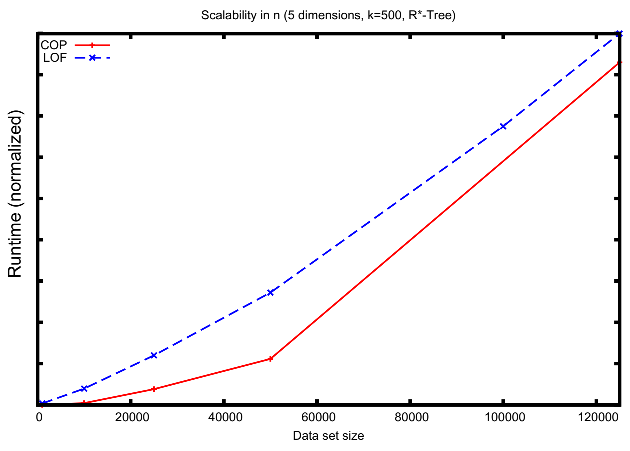
We studied the impact of the parameter k on the COP and the LOF values. We used a 3D data set with three generating mechanisms each exhibiting a different correlation and several points that considerably deviate from these generating mechanisms. Each point in the data set is labeled as a member of a generating mechanism, or as an outlier (cf. Figure 18.8(a)). We computed the COP value and the LOF value for different parameter settings and evaluated how accurate the outliers were identified using the well-known F-measure. The results are shown in Figure 18.8(b). In general, it can be observed, that both methods are rather robust around $15 \leq k \leq 20$ and degenerate significantly when clearly increasing k. In addition, the accuracy of COP is significantly above LOF in the range of a relevant parameter setting. The best F-measure achieved by COP is approximately 94% while LOF yields a maximum F-measure of approximately 88%. This is not surprising due to the characteristics of the data. This shows that COP is more suitable for data sets that show characteristics like the data set shown in Figure 18.8(a), appearing very likely in real-world data sets.

A last experiment compares COP with Stat on a 2D sample data set shown in Figure 18.9(a). Again, the accuracy of COP (cf. Figure 18.9(c)) is superior than the competing approach (cf. Figure 18.9(b)).

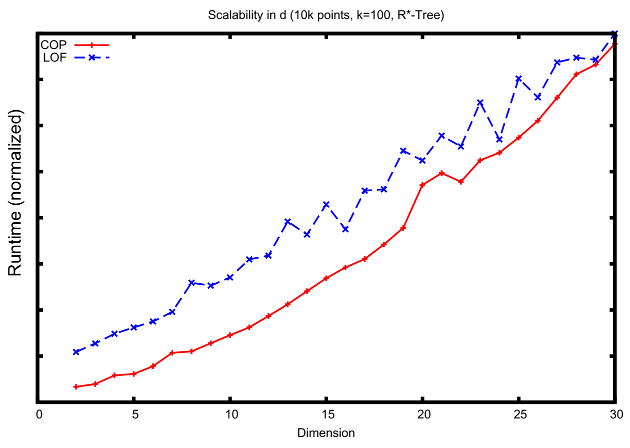
18.3.2 Scalability

In order to show the scalability of COP to large databases, we conducted a series of experiments where we fixed all but one of the free parameters including the database size n, the data dimensionality d and the number k of neighbors considered for the reference sets and altered the remaining parameter. We also used an R*-tree as underlying index-structure in all experiments.

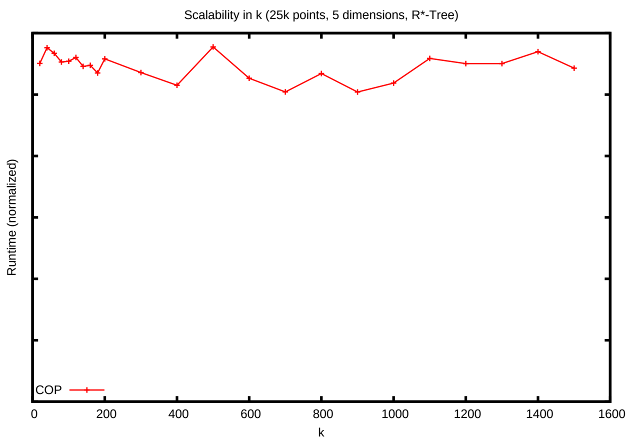
The runtime required to compute the COP value for all database objects w.r.t. increasing database size n in comparison to LOF is shown in Figure 18.10(a). Both approaches scale super-linear as expected. Let us note that we used materialized neighborhoods for LOF so that each neighborhood needs to



(a) Scalability w.r.t. n.



(b) Scalability w.r.t. d.



(c) Scalability w.r.t. k.

Figure 18.10: Scalability of COP.

be computed only once similar to COP. However, this implies a considerably higher storage overhead of LOF over COP.

We also examined the scalability of our novel outlier detection approach compared to LOF w.r.t. the dimensionality of the data points d. The results shown in Figure 18.10(b) suggest a linear scalability of both approaches. However, as discussed in the previous section, we expect COP to have a super-linear growth of runtime when increasing d.

Last but not least, we evaluated the impact of the parameter k on the runtime of the algorithm to determine the COP value of all database objects (Figure 18.10(c)). The results confirmed the theory that the algorithm for computing COP scales constant w.r.t. k.

In summary, our experiments have demonstrated that our novel outlier approach scales well to very large databases similar to established outlier models for large databases.

18.3.3 Results on Real-world Data

NBA data

We used COP to find outliers in a data set containing 15 statistical measures for 413 former and current NBA players obtained from the NBA website[1]. Features include the number of games played, the number of games in which the corresponding player appeared in the starting line-up, the number of minutes played per game, the number of points recorded per game, etc. We intentionally altered one arbitrarily selected record to produce a "measurement error" as follows. For the former player Danny Manning, we registered 83 games played and 398 appearances in the starting line-up which is obviously a contradiction[2].

Table 18.1 depicts the top three outliers according to COP. The highest-

[1]`http://www.nba.com`

[2]If Danny Manning or any supporter of Danny Manning reads this thesis, we apologize for our selection.

Table 18.1: Top outliers in the NBA data using COP.

Rank	**Player**	**COP**
1	Danny Manning	0,905
2	Jim McIlvaine	0,583015206
3	Dennis Rodman	0,561434904

ranked outlier is in fact the record featuring the measurement error ("Danny Manning"). The model explaining the outlier is rather interesting. The resulting error vector for this player indicates that he should have played 164 games more and should have appeared as starter in 178 games less. The second-ranked outlier is Jim McIlvaine an already retired center known to be a good shotblocker but a poor scorer. In fact the error vector of this player explains that exactly these characteristics qualified him as a significant outlier. The negative relationship between blocked shots and points per game is outstanding. The outlier with the third highest probability is Dennis Rodman, a retired power forward known to be a tremendous rebounder. However, the error vector of this player indicates an exceptionally high number of assists. This is interesting because usually, the big players like power forwards or centers are good rebounders but hand out only few assists. On the other hand, Rodman can be identified as a big man with an exceptionally high numbers of assists.

We also applied LOF on that data set. Even when trying to optimize the parameter *MinPts*, LOF could not find significant outliers. In all cases, the top outlier achieved a LOF value of below 1.8. This indicates that the objects in that data set exhibit a rather uniform density and outliers like the measurement error can only be detected when considering correlations as implemented by COP.

Table 18.2: Top outliers in the Wages data using COP.

Rank	COP	Description
1	0,911460762	low W despite high YE
2	0,880472682	low W despite high YE
3	0,879739261	high W despite low YE
4	0,856091566	low W despite high YE+YW
5	0,819392181	low W despite high YE+YW

Wages data

We further applied COP to the Wages data set[3] containing 534 entries from the 1985 Current Population Survey. Each data object features four attributes (A=age, YE=years of education, YW=years of work experience, and W=wage).

The top five outliers according to COP are listed in Table 18.2. All these outliers have very high scores. The two top-ranked objects are outliers because the corresponding persons have an exceptionally low wage compared to a rather large number of years of education. While there seems to be a rather large number of persons featuring a positive correlation among YE and W, these two outliers stand out due to a negative correlation among YE and W. The third-ranked outlier features also a negative correlation among YE and W but with contrary properties. The next two outliers exhibit a low wage although they have a high value in YE and YW, i.e. a long working experience and a sound education. This is contrary to another distinctive generating mechanism featuring a positive correlation among W and YE+YW. Again, LOF could not find significant outliers in that data set.

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

Part VI

Conclusions

Chapter 19

Summary

This thesis aimed at serving a twofold purpose. The first task was to systematically survey the very heterogenous field of clustering adaptations for high dimensional data. Second, we contributed new approaches and integrated them in the proposed systematics of algorithms and the reviewed problems typically encountered in the field.

Part I served to illustrate the background and motivation of the topic with the general context of data mining (Chapter 1) and some prominent application scenarios (Chapter 2).

Part II addressed the first main objective of the thesis in arranging the heterogeneous field of clustering algorithms for high dimensional data according to the typically addressed subproblems and approaches. To this end, Chapter 3 sketched the fundamental problem in clustering high dimensional data and characterized three different classes of clustering algorithms. The subsequent chapters discussed these different classes of algorithms in more detail. First, axis-parallel clustering with the consuetudinary but questionable categorization in *projected* and *subspace clustering* is surveyed in Chapter 4. Pattern-based clustering algorithms are surveyed in Chapter 5. Here, the point was not to give an exhaustive overview on existing approaches but to point out the relationship and differences in comparison to subspace and correlation clustering. The latter class of algorithms has been surveyed in

Chapter 6. Most of the approaches to this field have been developed by the author and constitute the second major contribution of the thesis. Chapter 7 concluded the systematic part with a discussion of the main problems and solutions. We have identified different aspects of the notorious "curse of dimensionality" that are addressed with different focus by different types of algorithms. So far, no efficient all-round-algorithm has been proposed and it seems unlikely to get one. It is not even clear whether such a solution would be desirable. Another open question, as pointed out in this systematic part, is the evaluation of different approaches. Since any approach uses its own assumptions and heuristics (and often even defines the objective in a different way), a comprehensive and fair experimental evaluation of a reasonable set of representatives for the different classes of solutions is not only missing so far but seems also a very demanding and complex task.

Contributions to the category of correlation clustering algorithms have been presented in Parts III–V. Part III collected adaptations of the density-based paradigm using PCA as a primitive to grasp correlated attributes and derive the corresponding arbitrarily oriented subspace. The first adaptation is the algorithm 4C (Chapter 8). A more robust, more efficient and more effective variant for flat correlation clustering is COPAC (Chapter 9). Hierarchical correlation clustering has been tackled by the approaches HiCO (Chapter 10) and ERiC (Chapter 11). For all correlation clustering algorithms based on PCA on a local selection of points, a framework to enhance the suitability of the selected set and the robustness of the applied PCA has been discussed in Chapter 12.

Nevertheless, as we have seen in Part IV, there remain weak points of all these density-based approaches applying PCA on a local selection of representative points (see Chapter 13). They rely on the so called *locality assumption* which is in view of high dimensional data rather naïve, as has also been discussed in Part II. Thus, as a global approach to correlation clustering, the algorithm CASH has been proposed and discussed in Chapter 14.

None of the existing correlation clustering algorithms derives a quantitative and qualitative model for each correlation cluster which is urgently needed in order to gain the full practical potentials from correlation cluster analysis. Part V described an original approach to derive quantitative information on the linear dependencies within correlation clusters. As discussed in Chapter 15, this step is not readily available for correlation clustering so far. The concepts for deriving quantitative and qualitative correlation clustering models described in Chapter 16 are independent of the clustering approach and can thus be applied as a post-processing step to any correlation clustering algorithm. The broad experimental evaluation demonstrated the beneficial impact of the proposed method on several applications of significant practical importance. It has been 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. Furthermore, as sample applications of the approach, Chapter 17 sketched how these quantitative models can be used to predict the probability distribution that an object is created by these models, and Chapter 18 described an adaptation of the approach to the outlier detection problem.

In Part VI, so far we summarized the contributions and results (Chapter 19) of the thesis. In the remaining Chapter 20, we will point out some open questions and possible directions for future work.

Chapter 20

Future Directions

At the end of this thesis, let us emphasize the potentials of the proposed methods for future research. For correlation clustering, future research could be guided by the following considerations:

- In the approaches discussed in this thesis, a deviation orthogonally to a correlation hyperplane has been accounted for based on thresholds or even using a refined Gaussian model of deviations. In this sense, the models discussed so far, tacitly assume the points being uniformly distributed within the correlation hyperplane. It could also be interesting to account for different distributions of the points within the correlation hyperplane, although this would lead to a considerably different intuition of the meaning of correlation clusters.
- There is no perfect solution for the assessment of the most suitable value of the correlation dimensionality of a given data set. Throughout the approaches discussed in this thesis, we used several different definitions of the correlation dimensionality. Naturally, all of them work differently well for different data sets. There are many aspects in this matter which are not discussed yet in depth. This would require some more analyses and could be another interesting aspect for future work.
- Another point of interest could be to define the correlation cluster mod-

els in a slightly more general way, allowing full dimensional clusters as a special case (as discussed for the outlier model in Chapter 18). This matter is also closely related to the assessment of the correlation dimensionality. In order to relax to the full dimensional case, a correlation dimensionality of 0 must be possible, i.e., no eigenvector is regarded as "strong".

- While much work has been done in identifying linear correlations among subsets of features in high dimensional data, the field of detecting non-linear correlations is largely unexplored. Using standard (linear) PCA, algorithms can detect linear correlations within a data set but fail in identifying non-linear structures. The concept of Kernel PCA [122] is a possibility 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 clustering to find non-linear correlation clusters in arbitrarily oriented subspaces. A related idea is to extend the derivation of a quantitative and qualitative model for linear correlation clusters to one for nonlinear correlation clusters. This problem could also be tackled by means of the Kernel PCA. Similarly, the Hough-transform is only a special case of many possible transformations. This field is also a resource of potential new approaches to non-linear correlation clustering.

Aside from these concrete issues, we would like to conclude this thesis with some informal final remarks.

It is a well accepted opinion that there is not such a thing like a general clustering technique suitable to all problems and universally applicable to arbitrary data sets. The aim of the concrete task of data analysis influences the choice of the clustering algorithm and obviously also the interpretation of the results of the clustering process. The appropriate choice of a clustering approach adequate to the problem at hand should be based on knowledge of the basic principles the particular clustering approach is based upon. Similarly, the interpretation of clustering results should be guided by the knowledge of the kinds of patterns a particular algorithm can or cannot find. In the survey on different approaches (Part II), we aimed at supporting such decisions and interpretations by a systematic overview on the different kinds of algorithms specialized to different problems known to occur in high dimensional data.

The family of axis-parallel subspace and projected clustering algorithms assumes that data objects belonging to the same cluster are near by each other but allows to assess the corresponding distance of objects w.r.t. subsets of the attributes due to the problem of increasingly poor separation of near and far points in higher dimensional data and the problem of irrelevant attributes. Pattern-based approaches often disregard the assumption, that a cluster consists of objects that are near by each other in the Euclidean space or some Euclidean subspace and, instead, aim at collecting objects following a similar behavioral pattern over a subset of attributes. These patterns relate to simple positive correlations among the considered attributes. Correlation clustering approaches generalize this approach to arbitrary complex positive or negative correlations but often assume, again, a certain density of the points in Euclidean space, too.

In identifying the problems addressed by the different families of algorithms and surveying the basic approaches pursued, we hope to stimulate further research. However, since different problems are addressed in different ways by the various approaches, questions arise whether more general solutions are possible addressing more problems in one approach. The presented systematic of different subspace clustering tasks — axis-parallel, pattern-based, arbitrarily-oriented, and here the aspects for future work: allowing also for search of non-linear correlations instead of the restriction to linear correlations — are subtly guided by this very same desire of finding the even more general solution that can tackle the even more general problem. This is, after all, a fundamental motivation in computer science: to describe a problem as generally as possible and to design a solution (if possible, a universal one) that tackles the original, concrete problem as a special case.

Accordingly, as a vision of data mining in the future, a more complex modeling of the world than mere feature vectors has been figured (cf. [87]).

The basic idea in such considerations is to make data mining approaches even more general and suitable to all kind of different data. This way, the complexity of real-world objects could be tackled in a more direct way compared to simplified representations by means of numerical attributes.

However, this quest for the always more general solution is a double-edged sword. Representing complex objects by means of simple objects like numerical feature vectors could also be understood as a way to incorporate domain knowledge into the data mining process and, as such, is a worthwhile goal. The domain expert seeks ways to use the important features of an object to e.g. classify new objects of the same type, eventually by employing sophisticated functions to transform attributes of some type to features of some other type. In the progress to generalized data mining one should not disregard of course the advances made so far. Incorporating domain knowledge fundamentally facilitates meaningful data mining.

But also restricting oneself to numerical data, it has been suggested that the more general approach would be preferable and also gain the better results since any structure imposed on the data would lead to biased results [63, 64]. On the first glance, this seems to make sense. On the other hand, it is unclear whether this really is what potential users need. More than that: For supervised learning it is well known that a bias-free learning is futile [101]. In our opinion, in the context of unsupervised learning it remains to be seen whether the more general approach is *theoretically* the better one.[1] *Practically*, however, the domain expert should decide which bias – if any – is most meaningful in a given application. Anyway, a clustering approach seems to be more useful if it is able to provide a model describing the clusters found. This allows to understand the grouping, to identify underlying mechanisms quantitatively, and, eventually, to refine scientific theories or target marketing strategies. What is more, while defining a model is not possible without defining the task properly, a properly defined task will help to develop an optimization approach based on a suitable model.

[1] There are even hints that the universal approach is impossible, see [49].

Unfortunately, practitioners of cluster analysis usually are not able to use the approach that suits their purpose best but only those approaches that are available in conveniently accessible statistical or data mining software systems. We totally agree with Kettenring [82] that a more close cooperation of developers and practitioners of cluster analysis in order "*to make sure that what is being offered represents best practices [...] would be a much more valuable contribution to cluster analysis than the next methodological advances*". So, as a final remark, let us express the opinion that focussing on old or new problems in developing new clustering approaches should even more closely analyze the needs of potential applicants. The ultimate goal of clustering is to find new knowledge by analyzing data sets describing some aspects of the world. And eventually, in these realms of applications, truth, as beauty, is in the eye of the beholder, or, as Hamlet said, "*for there is nothing either good or bad, but thinking makes it so*".

List of Figures

List of Tables

Bibliography

[1] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek. Robust clustering in arbitrarily oriented subspaces. In *Proceedings of the 8th SIAM International Conference on Data Mining (SDM), Atlanta, GA*, 2008.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] E. Achtert, H.-P. Kriegel, and A. Zimek. ELKI: a software system for evaluation of subspace clustering algorithms. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China*, 2008.

[9] C. C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory (ICDT), London, U.K.*, 2001.

[10] 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 ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA*, 1999.

[11] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional space. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX*, 2000.

[12] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Santa Barbara, CA*, 2001.

[13] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA*, 1998.

[14] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Minneapolis, MN*, 1994.

[15] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *Proceedings of the 6th European Conference on Principles of Knowledge Discovery and Data Mining (PKDD), Helsinki, Finland*, 2002.

[16] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA*, 1999.

[17] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR*, 1996.

[18] I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: dimensionality unbiased subspace clustering. In *Proceedings of the 7th International Conference on Data Mining (ICDM), Omaha, NE*, 2007.

[19] I. Assent, R. Krieger, E. Müller, and T. Seidl. VISA: visual subspace clustering analysis. *ACM SIGKDD Explorations Newsletter*, 9(2):5–12, 2007.

[20] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

[21] D. Barbara and P. Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Boston, MA*, 2000.

[22] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley&Sons, 3rd edition, 1994.

[23] S.D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, D.C.*, 2003.

[24] R. Bellman. *Adaptive Controll Processes. A Guided Tour.* Princeton University Press, 1961.

[25] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB), Zurich, Switzerland*, 1995.

[26] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Proceedings of the 6th Annual International Conference on Computational Molecular Biology (RECOMB), Washington, D.C.*, 2002.

[27] S. Berchtold, C. Böhm, H. V. Jagadish, H.-P. Kriegel, and J. Sander. Independent Quantization: An index compression technique for high-dimensional data spaces. In *Proceedings of the 16th International Conference on Data Engineering (ICDE), San Diego, CA*, 2000.

[28] S. Berchtold, C. Böhm, and H.-P. Kriegel. The Pyramid Technique: Towards breaking the curse of dimensionality. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Seattle, WA*, 1998.

[29] S. Berchtold, B. Ertl, D. A. Keim, H.-P. Kriegel, and T. Seidl. Fast nearest neighbor search in high-dimensional spaces. In *Proceedings of the 14th International Conference on Data Engineering (ICDE), Orlando, FL*, 1998.

[30] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-Tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB), Bombay, India*, 1996.

[31] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT), Jerusalem, Israel*, 1999.

[32] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[33] M. M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX*, 2000.

[34] 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.

[35] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France*, 2004.

[36] C. Böhm and H.-P. Kriegel. Dynamically optimizing high-dimensional index structures. In *Proceedings of the 7th International Conference on Extending Database Technology (EDBT), Konstanz, Germany*, 2000.

[37] C. Böhm and H.-P. Kriegel. Efficient construction of large high-dimensional indexes. In *Proceedings of the 16th International Conference on Data Engineering (ICDE), San Diego, CA*, 2000.

[38] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB), San Diego, CA*, 2000.

[39] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt*, 2000.

[40] 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.

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

[42] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Orlando, FL*, 2004.

[43] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Francisco, CA*, 2001.

[44] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma. Subspace clustering of high dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM), Orlando, FL*, 2004.

[45] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[46] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley&Sons, 2nd edition, 2001.

[47] 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.

[48] C. Faloutsos and I. Kamel. Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension. In *Pro-*

ceedings of the ACM International Conference on Management of Data (SIGMOD), Minneapolis, MN, 1994.

[49] C. Faloutsos and V. Megalooikonomou. On data mining, compression, and Kolmogorov complexity. *Data Mining and Knowledge Discovery*, 15(1):3–20, 2007.

[50] H. Fan, O. R. Zaïane, A. Foss, and J. Wu. A nonparametric outlier detection for efficiently discovering top-N outliers from engineering data. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore*, 2006.

[51] 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.

[52] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[53] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4):825–849, 2004.

[54] B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations.* Springer, 1999.

[55] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

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

[57] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences of the United States of America*, 97(22):12079–12084, 2000.

[58] 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.

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

[60] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2nd edition, 2006.

[61] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[62] R. Haralick and R. Harpaz. Linear manifold clustering. In *Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM), Leipzig, Germany*, 2005.

[63] R. Harpaz. *Model-Based Linear Manifold Clustering*. PhD thesis, The City University of New York, Department of Computer Science, 2007.

[64] R. Harpaz and R. Haralick. Linear manifold correlation clustering. *International Journal of Information Technology and Intelligent Computing*, 2(2), 2007.

[65] R. Harpaz and R. Haralick. Mining subspace correlations. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Honolulu, HI*, 2007.

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

[67] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, 2001.

[68] D. Hawkins. *Identification of Outliers*. Chapman and Hall, London, 1980.

[69] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt*, 2000.

[70] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD), New York City, NY*, 1998.

[71] P. V. C. Hough. Methods and means for recognizing complex patterns. U.S. Patent 3069654, December 18 1962.

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

[73] J. Ihmels, S. Bergmann, and N. Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[74] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 1999.

[75] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.

[76] W. Jin, A. K. H. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Singapore*, 2006.

[77] W. Jin, A.K. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Francisco, CA*, 2001.

[78] T. Johnson, I. Kwok, and R. Ng. Fast computation of 2-dimensional depth contours. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD), New York City, NY*, 1998.

[79] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[80] 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.

[81] N. Katayama and S. Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Tucson, AZ*, 1997.

[82] J. R. Kettenring. A perspective on cluster analysis. Short Communication. *Statistical Analysis and Data Mining*, 1(1):52–53, 2008.

[83] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York City, NY*, 1998.

[84] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), Edinburgh, Scotland*, 1999.

[85] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchthold. Efficient biased sampling for approximate clustering and outlier detection in large datasets. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1170–1187, 2003.

[86] F. Korn, B.-U. Pagel, and C. Falutsos. On the "dimensionality curse" and the "self-similarity blessing". *IEEE Transactions on Knowledge and Data Engineering*, 13(1):96–111, 2001.

[87] H.-P. Kriegel, K. M. Borgwardt, P. Kröger, A. Pryakhin, M. Schubert, and A. Zimek. Future trends in data mining. *Data Mining and Knowledge Discovery*, 15(1):87–97, 2007.

[88] 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.

[89] H.-P. Kriegel, P. Kröger, and A. Zimek. Detecting clusters in moderate-to-high dimensional data: Subspace clustering, pattern-based clustering, and correlation clustering. Tutorial at the 7th International Conference on Data Mining (ICDM), Omaha, NE, 2007. `http://www.ist.unomaha.edu/icdm2007/conference/ArthurZimekTutorial2.pdf%`.

[90] H.-P. Kriegel, Peer Kröger, E. Schubert, and A. Zimek. A general framework for increasing the robustness of PCA-based correlation clustering algorithms. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China*, 2008.

[91] J. Li, X. Huang, C. Selke, and J. Yong. A fast algorithm for finding correlation clusters in noise data. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Nanjing, China*, 2007.

[92] B. Liebl, U. Nennstiel-Ratzel, R. von Kries, R. Fingerhut, B. Olgemöller, A. Zapf, and A. A. Roscher. Very high compliance in an expanded MS-MS-based newborn screening program despite written parental consent. *Preventive Medicine*, 34(2):127–131, 2002.

[93] K. Lin, H. V. Jagadish, and C. Faloutsos. The TV-Tree: An index structure for high-dimensional data. *VLDB Journal*, 3:517–542, 1995.

[94] B. Liu, Y. Xia, and P. S. Yu. Clustering through decision tree construction. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM), Washington, D.C.*, 2000.

[95] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proceedings of the 23st International Conference on Data Engineering (ICDE), Istanbul, Turkey*, 2007.

[96] 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.

[97] 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.

[98] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[99] K. S. Miller. *Multidimensional Gaussian Distributions.* John Wiley&Sons, 1964.

[100] B. Mirkin. *Mathematical Classification and Clustering.* Kluwer, 1996.

[101] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[102] G. Moise, J. Sander, and M. Ester. P3C: A robust projected clustering algorithm. In *Proceedings of the 6th International Conference on Data Mining (ICDM), Hong Kong, China*, 2006.

[103] G. Moise, J. Sander, and M. Ester. Robust projected clustering. *Knowledge and Information Systems (KAIS)*, 14(3):273–298, 2008.

[104] T. M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the 8th Pacific Symposium on Biocomputing (PSB), Maui, HI*, 2003.

[105] H.S. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *Proceedings of the 1st SIAM International Conference on Data Mining (SDM), Chicago, IL*, 2001.

[106] B.-U. Pagel, F. Korn, and C. Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *Proceedings of the 16th International Conference on Data Engineering (ICDE), San Diego, CA*, 2000.

[107] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India*, 2003.

[108] E. Parros Machado de Sousa, C. Traina, A. Traina, and C. Faloutsos. How to use fractal dimension to find correlations between attributes. In *Proc. KDD-Workshop on Fractals and Self-similarity in Data Mining: Issues and Approaches*, 2002.

[109] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[110] 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.

[111] Y. Pei, O. Zaïane, and Y. Gao. An efficient reference-based approach to outlier detection in large datasets. In *Proceedings of the 6th International Conference on Data Mining (ICDM), Hong Kong, China*, 2006.

[112] J. Pfaltz. What constitutes a scientific database? In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM), Banff, Canada*, 2007.

[113] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Guissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[114] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI*, 2002.

[115] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX*, 2000.

[116] A. Rosenfeld. *Picture Processing by Computer.* Academic Press, 1969.

[117] P. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.

[118] 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.

[119] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[120] 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.

[121] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT), Valencia, Spain*, 1998.

[122] B. Schölkopf and A. J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[123] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl. 1):S243–S252, 2001.

[124] K. Sequeira and M. J. Zaki. SCHISM: a new approach to interesting subspace mining. *International Journal of Business Intelligence and Data Mining*, 1(2):137–160, 2005.

[125] Q. Sheng, Y. Moreau, and B. De Moor. Biclustering microarray data by Gibbs sampling. *Bioinformatics*, 19(Suppl. 2):ii196–ii205, 2003.

[126] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu. Mining maximal quasi-bicliques to co-cluster stocks and financial ratios for value investment. In *Proceedings of the 6th International Conference on Data Mining (ICDM), Hong Kong, China*, 2006.

[127] 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 Biology of the Cell*, 9:3273–3297, 1998.

[128] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Montreal, Canada*, 1996.

[129] P. Sun and S. Chawla. On local spatial outliers. In *Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, U.K.*, 2004.

[130] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison Wesley, 2006.

[131] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl. 1):S136–S144, 2002.

[132] A. Tanay, R. Sharan, and R. Shamir. Biclustering algorithms: A survey. In S. Aluru, editor, *Handbook of Computational Molecular Biology.* Chapman & Hall, 2006.

[133] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Taipei, Taiwan*, 2002.

[134] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[135] J. Tukey. *Exploratory Data Analysis.* Addison-Wesley, 1977.

[136] A. K. H. Tung, X. Xu, and C. B. Ooi. CURLER: Finding and visualizing nonlinear correlated clusters. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Baltimore, ML*, 2005.

[137] I. Van Mechelen, H.-H. Bock, and P. De Boeck. Two-mode clustering methods: a structured overview. *Statistical methods in medical research*, 13:363–394, 2004.

[138] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI*, 2002.

[139] 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.

[140] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York City, NY*, 1998.

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

[142] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[143] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th International Conference on Data Engineering (ICDE), Orlando, FL*, 1998.

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

[145] K. Y. Yip, D. W. Cheung, and M. K. Ng. HARP: a practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1387–1397, 2004.

[146] K. Y. Yip, D. W. Cheung, and M. K. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In *Proceedings of the 21st International Conference on Data Engineering (ICDE), Tokyo, Japan*, 2005.

[147] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the 3th International Conference on Data Mining (ICDM), Melbourne, FL*, 2003.

[148] M. L. Yiu and N. Mamoulis. Iterative projected clustering by subspace mining. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):176–189, 2005.

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

[150] C. Zhu, H. Kitagawa, and C. Faloutsos. Example-based robust outlier detection in high dimensional datasets. In *Proceedings of the 5th International Conference on Data Mining (ICDM), Houston, TX*, 2005.

Acknowledgements

While I was working on this thesis I received a lot of support and encouragement. I am very grateful for all the help I got during the last few years that did pass by far too fast. Unfortunately, I can mention only some of all these adjuvant people, but my sincere thanks are dedicated, 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. Hans-Peter Kriegel. He initiated and supported this work with his great experience and with providing the organizational background and gave me the opportunity to work on this challenging domain. It is not a secret that he is especially able to create an inspiring and supportive working atmosphere within his database research group. Furthermore, I warmly thank Prof. Dr. Thomas Seidl 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 I will not forget all the fun we had. In particular, I want to thank Dr. Elke Achtert, Johannes Aßfalg, Thomas Bernecker, Prof. Dr. Christian Böhm, Dr. Karsten Borgwardt, Dr. Stefan Brecheisen, Tobias Emrich, Franz Graf, Dr. Karin Kailing, Dr. Peer Kröger, Dr. “Pete the Weasle” Peter Kunath, Dr. Alexey Pryakhin, Dr. Matthias Renz, Dr. Matthias Schubert, Marisa Thoma, Steffi

Wanka, and Andreas Züfle for constructive and productive team-work, as well as for many helpful discussions.

The background help of Susanne Grienberger was another reason that working in Hans-Peter Kriegel's group was that comfortable. Her support in managing the administrative burden and in preparing manuscripts as well as teaching material was invaluable.

Furthermore, I want to express special thanks to Franz "Sherlock" Krojer, who helped to master all technical issues. He promptly provided tools that helped to accelerate my work – and if these tools ever did unexpectedly not work, his eagerness to clarify the issue was always a pleasure.

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. Productive discussions contributed to further development of the ideas I tried to lay out in this thesis.

Last but not least, I like to express my deepest gratitude to my family and my friends for their support and encouragement during the time I was engaged in this study. I regret having been unavailable and absentminded often in the last years and I am much obliged for all their persistent truth and faithfulness.

Arthur Zimek

Munich, July 2008

Curriculum Vitae

Arthur Zimek was born on August 18, 1971 in Nuremberg, Germany. He attended primary school from 1979 to 1982, and high-school from 1982 to 1991.

From July 1991 until September 1992, he served in the mandatory civil service at the hospital *Marienhospital Darmstadt*, Germany.

He studied Theology at the Universities Mainz and Innsbruck from 1992 to 1997, graduating with a diploma degree in Theology in 1997.

He studied Philosophy at the *Hochschule für Philosophie* in Munich from 1997 to 2000, graduating with the degree Magister Artium in Philosophy.

He entered the *Ludwig-Maximilians-Universität München* (LMU) in 2000, studying Bioinformatics. His diploma thesis was on "*Hierarchical Classification Using Ensembles of Nested Dichotomies*", supervised by Prof. Dr. Stefan Kramer (TUM). He graduated with the diploma degree in Bioinformatics in 2005.

In May 2005, Arthur Zimek started working at the LMU as a research and teaching assistant in the group of Prof. Dr. Hans-Peter Kriegel, the chair of the teaching and research unit for database and information systems at the Department “Institute for Computer Science”. His research interests include knowledge discovery in databases, machine learning and data mining and the application scenario of computational biology and bioinformatics.