Coping With New Challenges for Density-Based Clustering

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

von

Peer Kröger

Tag der Einreichung: 18.5.2004
Tag der mündlichen Prüfung: 8.7.2004

Berichterstatter:
Prof. Dr. Hans-Peter Kriegel, Ludwig-Maximilians-Universität München
Prof. Dr. Jörg Sander, University of Alberta (Kanada)

Acknowledgement

I would like to thank all the people for supporting me during the past years and in particular during the development of this thesis, even if I cannot mention all of their names here.

First of all, I extend my warmest thanks to my supervisor, Prof. Dr. Hans-Peter Kriegel who initiated and supported this work with his long standing experiences and the organizational background. He made this work possible by taking special care of an inspiring and supportive working environment in the database research group. He always encouraged me in hard times and also let me any freedom for my research activities.

I also want to thank Prof. Dr. Jörg Sander for his interest in my work. He was kindly willing to act as second referee to this work and provided me with several fruitful discussions and valuable hints for the completion of this thesis.

Parts of this work has been performed within the joint project BFAM (Bioinformatics for the Functional Analysis of Mammalian genomes) funded by the German Ministry for Education, Science, Research and Technology (BMBF) under grant number 031U112F and embedded in the German Genome Analysis Network (NGFN).

This work would not have been initiated and matured without the cooperation of and discussion with my colleagues in the database research group. In particular, I want to thank Karin Kailing, Martin Pfeifle, Matthias Schubert, Stefan Brecheisen, Alexey Pryakhin, Prof. Dr. Christian Böhm, Prof. Dr. Thomas Seidl, and Dr. Marco Pötke for constructive and productive team-

work, as well as Matthias Renz, Eshref Januzaj, Peter Kunath, and Stefan Schönauer for many helpful discussions. Especially, I also want to mention Stefan Schönauer for his administrative advice and support and for reading several parts of this thesis to give some valuable hints for improvements.

For such a practically oriented work as this thesis, a large amount of implementation, data preprocessing, and testing were necessary. I whish to thank all the students that helped me to manage the various tasks, in particular, Arthur Zimek, Elke Achtert, Stefanie Wanka, Heribert Mühlberger, and Maximilian Viermetz.

This thesis could not have been completed without the tremendous background support of Susanne Grienberger. Besides bearing much of the administrative burdens, she carefully read this thesis and gave my invaluable hints for improving the language. In addition, I want to express special thanks to Franz Krojer for his openness and flexibility meeting my sometimes strange technical demands and problems and for being on the spot when my workstation crashed.

Last but not least, I want to thank my family. My parents who always supported my career and encouraged me to find my way, and my fiancée, Cornelia Bucher, for all the love and care she gave me, for her patience enduring my absent-mindedness and for many sacrifices she shouldered in the last months.

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 tasks 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. Beside many others, the density-based clustering notion underlying the algorithm DBSCAN and its hierarchical extension OPTICS has been proposed recently, being one of the most successful approaches to clustering.

In this thesis, our aim is to advance the state-of-the-art clustering, especially density-based clustering by identifying novel challenges for density-based clustering and proposing innovative and solid solutions for these challenges.

We describe the development of the industrial prototype BOSS (Browsing OPTICS plots for Similarity Search) which is a first step towards developing a comprehensive, scalable and distributed computing solution designed to make the efficiency and analytical capabilities of OPTICS available to a broader audience. For the development of BOSS, several key enhancements of OPTICS are required which are addressed in this thesis. We develop incremental algorithms of OPTICS to efficiently reconstruct the hierarchical clustering structure in frequently updated databases, in particular, when a set of objects is inserted in or deleted from the database. We empirically show

that these incremental algorithms yield significant speed-up factors over the original OPTICS algorithm. Furthermore, we propose a novel algorithm for automatic extraction of clusters from hierarchical clustering representations that outperforms comparative methods, and introduce two novel approaches for selecting meaningful representatives, using the density-based concepts of OPTICS and producing better results than the related medoid approach.

Another major challenge for density-based clustering is to cope with high dimensional data. Many today's real-world data sets contain a large number of measurements (or features) for a single data object. Usually, global feature reduction techniques cannot be applied to these data sets. Thus, the task of feature selection must be combined with and incooperated into the clustering process. In this thesis, we present original extensions and enhancements of the density-based clustering notion to cope with high dimensional data. In particular, we propose an algorithm called SUBCLU (density based SUBspace CLUstering) that extends DBSCAN to the problem of *subspace clustering*. SUBCLU efficiently computes all clusters that would have been found if DBSCAN is applied to all possible subspaces of the feature space. An experimental evaluation on real-world data sets illustrates that SUBCLU is more effective than existing subspace clustering algorithms because it is able to find clusters of arbitrary size and shape, and produces determine results. A semi-hierarchical extension of SUBCLU called RIS (Ranking Interesting Subspaces) is proposed that does not compute the subspace clusters directly, but generates a list of subspaces ranked by their clustering characteristics. A hierarchical clustering algorithm can be applied to these interesting subspaces in order to compute a hierarchical (subspace) clustering. A comparative evaluation of RIS and SUBCLU shows that RIS in combination with OPTICS can achieve an information gain over SUBCLU. In addition, we propose the algorithm 4C (Computing Correlation Connected Clusters) that extends the concepts of DBSCAN to compute density-based correlation clusters. 4C benefits from an innovative, well-defined and effective clustering model, outperforming related approaches in terms of clustering quality on real-world data sets.

Zusammenfassung

Knowledge Discovery in Databases (KDD) ist der Prozess der (semi-)automatischen Extraktion von Wissen aus Datenbanken, das gültig, bisher unbekannt und potentiell nützlich für eine gegebene Anwendung ist. Der zentrale Schritt des KDD-Prozesses ist das Data Mining. Eine der wichtigsten Aufgaben des Data Mining ist 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. Das dichtebasierte Clustermodell und die darauf aufbauenden Algorithmen DBSCAN und OPTICS sind unter einer Vielzahl anderer Clustering-Ansätze eine der erfolgreichsten Methoden zum Clustering.

Im Rahmen dieser Dissertation wollen wir den aktuellen Stand der Technik im Bereich Clustering und speziell im Bereich dichtebasiertes Clustering voranbringen. Dazu erarbeiten wir neue Herausforderungen für das dichtebasierte Clustermodell und schlagen dazu innovative Lösungen vor.

Zunächst steht die Entwicklung des industriellen Prototyps BOSS (Browsing OPTICS plots for Similarity Search) im Mittelpunkt dieser Arbeit. BOSS ist ein erster Beitrag zu einer umfassenden, skalierbaren und verteilten Softwarelösung, die eine Nutzung der Effizienzvorteile und die analytischen Möglichkeiten des dichtebasierten, hierarchischen Clustering-Algorithmus OPTICS für ein breites Publikum ermöglichen. Zur Entwicklung von BOSS werden drei entscheidende Erweiterungen von OPTICS benötigt: Wir entwickeln eine inkrementelle Version von OPTICS um nach einem Update der Datenbank (Einfügen/Löschen einer Menge von Objekten) die hierarchische Clustering Struktur effizient zu reorganisieren. Anhand von Experimenten mit

synthetischen und realen Daten zeigen wir, dass die vorgeschlagenen, inkrementellen Algorithmen deutliche Beschleunigungsfaktoren gegenüber dem originalen OPTICS-Algorithmus erzielen. Desweiteren schlagen wir einen neuen Algorithmus zur automatischen Clusterextraktion aus hierarchischen Repräsentationen und zwei innovative Methoden zur automatischen Auswahl geeigneter Clusterrepräsentaten vor. Unsere neuen Techniken erzielen bei Tests auf mehreren realen Datenbanken im Vergleich zu den konkurrierenden Verfahren bessere Ergebnisse.

Eine weitere Herausforderung für Clustering-Verfahren stellen hochdimensionale Featureräume dar. Reale Datensätze beinhalten dank moderner Verfahren zur Datenerhebung häufig sehr viele Merkmale. Teile dieser Merkmale unterliegen oft Rauschen oder Abhängigkeiten und können meist nicht im Vorfeld ausgesiebt werden, da diese Effekte jeweils in Teilen der Datenbank unterschiedlich ausgeprägt sind. Daher muss die Wahl der Features mit dem Data-Mining-Verfahren verknüpft werden. Im Rahmen dieser Arbeit stellen wir innovative Erweiterungen des dichtebasierten Clustermodells für hochdimensionale Daten vor. Wir entwickeln SUBCLU (dichtebasiertes SUBspace CLUstering), ein auf DBSCAN basierender Subspace Clustering Algorithmus. SUBCLU erzeugt effizient alle Cluster, die gefunden werden, wenn man DBSCAN auf alle möglichen Teilräume des Datensatzes anwendet. Experimente auf realen Daten zeigen, dass SUBCLU effektiver als vergleichbare Algorithmen ist. RIS (Ranking Interesting Subspaces), eine semi-hierarchische Erweiterung von SUBCLU, wird vorgeschlagen, das nicht mehr direkt die Teilraumcluster berechnet, sondern eine Liste von Teilräumen geordnet anhand ihrer Clustering-Qualität erzeugt. Dadurch können hierarchische Partitionierungen auf ausgewählten Teilräumen erzeugt werden. Experimente belegen, dass RIS in Kombination mit OPTICS ein Informationsgewinn gegenüber SUBCLU erreicht. Außerdem stellen wir den neuartigen Korrelationscluster Algorithmus 4C (Computing Correlation Connected Clusters) vor. 4C basiert auf einem innovativen und wohldefinierten Clustermodell und erzielt in unseren Experimenten mit realen Daten bessere Ergebnisse als vergleichbare Clustering-Ansätze.

Survey of Chapters

I Preliminaries **3**

1 Introduction 3

2 Density-Based Clustering 13

II Using Density-Based Hierarchical Clustering for Similarity Search Applications **33**

3 A Browsing Tool for Similarity Search 33

4 Incremental Clustering 43

5 Cluster Recognition and Representation 67

6 BOSS: Browsing OPTICS Plots for Similarity Search 87

III Adopting Density-Based Clustering to High Dimensional Data **99**

7 Clustering High Dimensional Data 99

8 Subspace Clustering 113

9 Correlation Clustering 151

IV Conclusions **185**

10 Summary and Future Directions 185

Contents

Acknowledgement iv

Abstract vi

Zusammenfassung viii

Survey of Chapters ix

I Preliminaries 1

1 Introduction 3

1.1 Knowledge Discovery in Databases, Data Mining and Clustering 4

1.2 Outline of the Thesis 9

2 Density-Based Clustering 13

2.1 General Clustering Approaches 14

2.1.1 Partitioning Algorithms 14

2.1.2 Hierarchical Algorithms 16

2.2 Basic Notations 17

2.3 Foundations of Density-Based Clustering 18

2.3.1 Clusters as Density Connected Sets 19

2.3.2 Density-Based Hierarchical Decompositions 24

II Using Density-Based Hierarchical Clustering for Similarity Search Applications 31

3 A Browsing Tool for Similarity Search 33

3.1 Motivation . . . 34

3.1.1 Visual Data Mining . . . 35

3.1.2 Similarity Search . . . 36

3.1.3 Evaluation of Similarity Models . . . 38

3.2 Required Enhancements . . . 39

4 Incremental Clustering 43

4.1 Related Work . . . 44

4.2 Incremental OPTICS . . . 45

4.2.1 General Ideas and Concepts . . . 46

4.2.2 Incremental Insertion of a Point . . . 52

4.2.3 Incremental Deletion of a Point . . . 58

4.2.4 Extensions for Bulk Updates . . . 61

4.3 Experimental Evaluation . . . 61

4.4 Summary . . . 64

5 Cluster Recognition and Representation 67

5.1 Cluster Recognition . . . 68

5.1.1 Related Work . . . 68

5.1.2 Gradient Clustering . . . 70

5.1.3 Experimental Evaluation . . . 74

5.2 Cluster Representation . . . 77

5.2.1 The Minimum Core Distance Approach . . . 78

5.2.2 The Maximum Successors Approach . . . 79

5.2.3 Experimental Evaluation . . . 82

5.3 Summary . . . 84

6 BOSS: Browsing OPTICS Plots for Similarity Search 87

6.1 System Architecture 88

6.2 Sample Applications 90

6.2.1 Visual Data Mining 90

6.2.2 Evaluation of Similarity Models 91

6.3 Summary and Discussion 95

III Adopting Density-Based Clustering to High Dimensional Data 97

7 Clustering High Dimensional Data 99

7.1 The Curse of Dimensionality 100

7.2 General Approaches for Clustering High Dimensional Data . . 102

7.3 Sample Applications 106

7.3.1 Gene Expression Analysis 107

7.3.2 Metabolic Screening of Newborns 110

7.4 Summary 111

8 Subspace Clustering 113

8.1 Related Work 114

8.2 Foundations of Density-Based Subspace Clustering 117

8.2.1 Adapting Density-Based Concepts to Subspace Clustering 117

8.2.2 Monotonicity Properties 120

8.3 Density-Based Subspace Clustering 124

8.3.1 General Idea 124

8.3.2 Algorithm SUBCLU 126

8.3.3 Experimental Evaluation 129

8.4 Density-Based Subspace Ranking 133

8.4.1 Motivation 133

8.4.2 General Idea . . . 135
8.4.3 Ranking Interesting Subspaces . . . 136
8.4.4 Algorithm RIS . . . 139
8.4.5 Experimental Evaluation . . . 144
8.5 Summary and Discussion . . . 148

9 Correlation Clustering 151
9.1 Motivation and Related Work . . . 152
9.2 Foundations of Connected Correlation Clustering . . . 155
9.2.1 Correlation Sets . . . 155
9.2.2 Clusters as Correlation Connected Sets . . . 157
9.3 Computing Correlation Connected Clusters . . . 164
9.3.1 Algorithm 4C . . . 164
9.3.2 Complexity Analysis . . . 166
9.3.3 Input Parameters . . . 167
9.4 Quality Evaluation . . . 168
9.5 Modifications and Specializations . . . 174
9.5.1 A Variant for Pattern-Based Clustering . . . 174
9.5.2 A Variant for Projected Clustering . . . 177
9.6 Summary . . . 180

IV Conclusions 183

10 Summary and Future Directions 185
10.1 Summary of Contributions . . . 186
10.1.1 Preliminaries (Part I) . . . 186
10.1.2 Using Density-Based Hierarchical Clustering for Similarity Search Applications (Part II) . . . 186
10.1.3 Adopting Density-Based Clustering to High Dimensional Data (Part III) . . . 188

10.2 Future Work 189

List of Figures **193**

List of Tables **196**

References **197**

Part I

Preliminaries

Chapter 1

Introduction

Today's capabilities of data generation produce larger and larger amounts of data that are collected and stored in databases. Knowledge Discovery in Databases (KDD) is an interdisciplinary field, aimed at extracting valuable knowledge from such large databases. At the core of the KDD process is the Data Mining step which embraces many data mining methods. One of them is Clustering, the central topic of this thesis. In this chapter, the KDD process is introduced and discussed in detail. Then we describe the data mining step in more detail, and review the most important and influential methods of data mining. In this thesis, we focus on clustering, in particular on the so-called *density-based* clustering approach. We identify several novel challenges for this density-based clustering approach, solutions of which are proposed in this thesis. The chapter concludes with an outline of this thesis.

1.1 Knowledge Discovery in Databases, Data Mining and Clustering

With steadily advancing capabilities of both generating and collecting data in the last several decades, a tremendous amount of information is available in nearly all different aspects of life. These mountains of stored data contain information from such diverse sources as credit card transactions, telephone calls, space observatories, genome research, gene expression profiles, supermarket purchase transactions (market basket data) or web clickstreams. The information hidden in such data is mostly of outstanding strategic and financial importance for companies or may enable scientific breakthroughs. However, without the help of automated analysis tools, the full use cannot be made out of this mass of data.

Knowledge Discovery in Databases (KDD) is an interdisciplinary field bringing together techniques from various areas, e.g. machine learning, statistics, databases and visualization, to address the issues of analyzing such huge data sets, and extracting knowledge from them.

Classical techniques from the areas of statistics and on-line analytical processing (OLAP) were not designed to cope with today's large databases and the new demands on the power of the analysis method. Important reasons for the limited applicability of these methods include the following:

- **Massive data sets:** they do not scale well with large data sets, i.e. a large number of records and/or a large number of dimensions/attributes. Many statistical packages assume that the whole data set can be "loaded" into main memory.
- **Hypothesis-verification vs. exploratory data analysis (EDA):** statistical techniques are primarily focused on the verification of user-defined hypothesis, while many current problems demand the possibility to analyze the data by exploration.
- **Visualization:** OLAP techniques are well suited for visualizing the entire data set or relatively simple aggregates and groupings of the

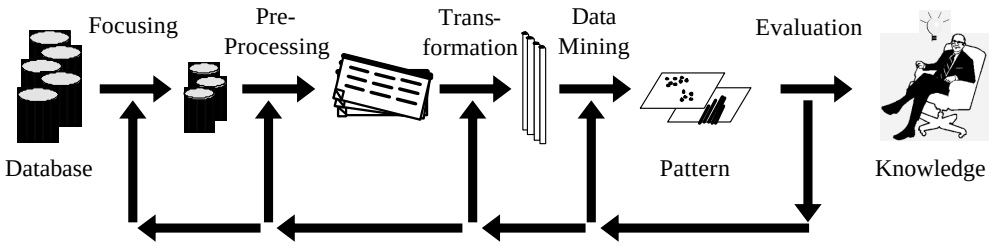


Figure 1.1: The KDD process.

data records. However, they cannot visualize complex patterns hidden deep inside the data.

- **Storage and access:** Most OLAP and statistical techniques require easy access to the whole data set. Many data warehouses and databases, however, are located on remote servers, so access to the entire data set may not be allowed or simply not possible because of network traffic constraints.

To meet these new requirements and constraints of massive data sets, novel methods, algorithms, and techniques have been developed in the new research area of KDD. In [FPSS96b] the following definition is proposed:

Knowledge Discovery in Databases *is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

Figure 1.1 gives a detailed overview of the KDD process, showing the basic flow of steps. Frequently, multiple iterations among these steps are necessary.

1. **Focus:** Create a target data set by selecting a subset of the attributes (projection) and/or records (sampling) of the database.

2. **Preprocessing:** Clean the data, i.e., for example, remove noise, model

or account for noise, decide on suitable strategies to handle missing attribute values or add derived features.

3. **Transformation:** Reduce the data further, e.g. by using dimensionality reduction to minimize the effective number of attributes, by finding invariant representations of the data or by identifying useful features to represent the data depending on the goal of the discovery task.

4. **Data Mining:** Match the goal of the knowledge discovery task with a suitable data mining method, e.g. classification. Choose the data mining algorithm for searching patterns in the data (e.g. a support vector machine or a decision tree like C4.5). Finally, apply the chosen algorithm to the transformed data set, in order to receive a set of patterns extracted from the data.

5. **Evaluation:** Interpret the mined patterns, e.g. by visually representing the patterns and/or the subsets of the data. This may result in returning to one of the previous steps (if the discovered knowledge is unsatisfactory or new insights are gained, making further investigations necessary), and leading to an iterative process. Finally, consolidate the discovered knowledge, e.g. by documenting and reporting it or taking suitable actions.

For a survey of industrial applications of KDD see [PSBK+96], and for applications in science data analysis and research see [FPSS96a].

The core step in the KDD process is the application of a data mining algorithm. Thus, the notions "KDD" and "data mining" are often used interchangeably. In [FPSS96b], data mining is defined as follows.

Data mining *is a step in the KDD process consisting of applying data analysis algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data.*

In [HK01] the diverse data mining algorithms proposed recently in the

literature are classified according to the following primary data mining methods:

- **Clustering:** group the objects of a database into clusters by maximizing the intra-cluster similarity and minimizing the inter-cluster similarity.
- **Outlier Detection:** find outliers, i.e. data objects that do not correspond to the general behavior or model of the data.
- **Classification/Prediction:** learn a function or model to classify a data object into one of several predefined classes.
- **Association Analysis:** find association rules that show attribute-value conditions that occur frequently together in the database.
- **Evolution Analysis:** describe and model regularities or trends for objects whose behavior changes over time.
- **Characterization and Discrimination:** summerize general characteristics or features of a subset of the database and compare particular subsets of the data with comparative subsets.

In this thesis we focus on clustering.

Clustering *is the task of grouping objects of a database into classes (clusters) such that objects within one cluster are most similar to each other and objects of different clusters are most dissimilar to each other.*

Thus, clustering aims at detecting new classes of data without any *a priori* knowledge. Therefore, clustering is often also called *unsupervised learning* in contrast to classification where the classes are predefined and which is often also called *supervised learning*.

The task of clustering has been studied in statistics (e.g. [McQ67], [Har75], [JD88]), machine learning (e.g. [CKS+88], [Fis95], [FPL91]), and more recently databases (e.g. [NH94], [ZRM96], [SEKX98]). The reason for

the new database-oriented approaches have already been indicated above. Well-known clustering algorithms from statistics such as k-means [McQ67] or SLINK [Sib73] do not scale well with large databases. In addition, the entire database is assumed to reside in main memory at the same time during the clustering process. Furthermore, novel real-world database applications create new challenges for clustering algorithms including — among others:

- **Incremental maintenance of mined patterns:**
  In a dynamic database environment, updates such as insertion or deletion of data objects may frequently occur. The necessary update of the mined clustering structure should be worked out incrementally for efficiency, i.e. availability, reasons.

- **Usability of clustering results for semi-automatic cluster analysis and further postprocessing:**
  Solid cluster extraction from hierarchical cluster representations is a mandatory assumption for semi-automatic cluster analysis and postprocessing. Meaningful cluster representatives form the basis to get a quick overview of the generated clusters. Both cluster extraction and cluster representation can help to get a quick overview of massive data sets.

- **Clustering in high dimensional feature spaces:**
  Most clustering algorithms work on feature databases, i.e. databases of points in some d-dimensional space, using the proximity of points, e.g. the Euclidean distance, as a measure of (dis)similarity. Usually, these algorithms compute clusters in the "full" (d-) dimensional space, i.e. all features are taken into account for computing the distance between two points. However, as indicated above, nowadays more and more data sets are high dimensional, i.e. provide a huge number of features. In high dimensional spaces, traditional clustering algorithms tend to break down in terms of efficiency as well as accuracy because data do not cluster well anymore. Therefore, high dimensional feature spaces require new clustering concepts to cope with today's data.

In this thesis, we will address these three novel challenges in the context of density-based clustering.

1.2 Outline of the Thesis

The starting point of this thesis is the density-based clustering approach, in particular the concepts of density connected clusters underlying the algorithms DBSCAN [EKSX96] and its hierarchical extension OPTICS [ABKS99]. In this thesis, we propose techniques to cope with the three challenges mentioned in the previous section in the context of density-based clustering. The major contributions of this thesis include:

1. An incremental version of the OPTICS algorithm to efficiently maintain the computed clusters in large, dynamic databases.
2. A new algorithm for solid cluster extraction from hierarchical cluster representations computed by OPTICS and innovative methods for selecting intuitive cluster representatives for the recognized clusters.
3. The techniques mentioned in 1. and 2. are incorporated into an industrial prototype called BOSS (Browsing OPTICS plots for Similarity Search) that uses density-based hierarchical clustering for advanced similarity search purposes.
4. Original adoptions and extensions of the density-based clustering concepts to detect density connected clusters in high dimensional feature spaces.
5. A novel correlation clustering algorithm based on a combination of density-based clustering concepts with principal component analysis (PCA).

The remainder of this thesis is organized as follows.

Chapter 2 provides a brief and rather general overview over existing clustering algorithms and introduces the density-based clustering notion underlying DBSCAN and OPTICS. As mentioned above, the concepts described in this chapter form the basis of the techniques proposed in this thesis. In addition, the chapter describes some basic notations used throughout this thesis.

Part II describes the above mentioned prototype BOSS and all concepts necessary for its development.

Chapter 3 motivates the use of density-based hierarchical clustering for advanced similarity search purposes. In particular, we sketch our interactive data browsing tool BOSS and outline three of its application ranges. The chapter concludes with a list of improvements to density-based hierarchical clustering required for the development of BOSS.

Chapter 4 deals with the efficient update of the hierarchical clustering structure in a dynamic database. First, related work is reviewed. Then, the concepts underlying the clustering algorithm OPTICS are extended to enable the development of an incremental version. An efficient, incremental version of the OPTICS algorithm called *IncOPTICS* is proposed thereafter. A performance evaluation using synthetic and real-world data sets is presented, showing that IncOPTICS yields a significant speed-up over OPTICS.

Chapter 5 addresses the tasks of solid cluster recognition from hierarchical clustering structures and the selection of meaningful cluster representatives. First, recent approaches for cluster extraction are discussed and a novel approach called *Gradient Clustering* is proposed. A comparative experimental evaluation based on real-world data sets is presented. Second, two original approaches for selecting cluster representatives are introduced and empirically evaluated.

Chapter 6 describes details of the browsing tool BOSS. In particular, some details about the implementation of the BOSS prototype are presented and two sample applications of BOSS are illustrated, including an application to visual data mining and semi-automatic cluster analysis, as well as an

application to the evaluation of different similarity models.

Part III presents innovative adoptions and extensions necessary to compute density-based clusters in high dimensional feature spaces.

Chapter 7 introduces general aspects of clustering high dimensional data. First, the major problems associated with high dimensional feature spaces in the context of clustering which are often summarized by the term *curse of dimensionality* are worked out. Then, a classification of general approaches to find clusters in high dimensional data is presented. The chapter concludes with two motivating examples of real-world applications, where clustering high dimensional feature spaces is required. It is also investigated which general approach is suited for the two sample applications and a test bed of real-world data sets is described. The methods proposed in this part of the thesis are evaluated using these data sets.

Chapter 8 presents two novel algorithms that address the subspace clustering problem. First, we adopt the density-based clustering concepts to the problem of subspace clustering and explore monotonicity properties of these concepts in order to investigate opportunities for efficient subspace searching. Based on these considerations, we present SUBCLU (density-based SUBspace CLUstering), a density-based subspace clustering algorithm. A broad experimental evaluation of SUBCLU shows its superior accuracy compared to existing subspace clustering algorithms. After that, an extension of SUBCLU called RIS (Ranking Interesting Subspaces for clustering) is proposed. A comparative evaluation of RIS and SUBCLU reveals that RIS can gain significantly more information than SUBCLU.

Chapter 9 proposes a novel correlation clustering algorithm called *4C* which is based on a combination of density-based clustering and principal component analysis (PCA). First, the concept of *correlation connected clusters* is formalized. Then, we present the details of the 4C algorithm. An experimental evaluation compares 4C to several competing clustering algorithms showing its superior performance. After that, some modifications and specializations are introduced, illustrating how the concepts of 4C can

be adopted to the pattern-based clustering and the projected clustering approach.

Part IV concludes this thesis.

Chapter 10 summerizes and discusses the major contributions of the thesis. It concludes with indicating some potentials for possible future research directions.

Chapter 2

Density-Based Clustering

Many clustering algorithms have been proposed recently. This thesis will base on the density-based clustering approach which turned out to be one of the most effective and also efficient one. In this chapter, we will first give a brief and rather general overview and classification of recently proposed clustering algorithms (cf. Section 2.1) and establish basic notations used throughout this thesis in Section 2.2. After that, we give a detailed introduction to the density-based notion of clusters (cf. Section 2.3). In particular, we introduce the notion of *flat density connected sets* as proposed in [EKSX96] providing the basis of the algorithm DBSCAN and discuss the hierarchical extensions leading to the concept of *density-based cluster orderings* as proposed in [ABKS99] constituting the foundations of the algorithm OPTICS.

2.1 General Clustering Approaches

In the past decade, many algorithmic solutions for the problem of clustering have been proposed. In the following, we will present a general classification of these approaches. In particular, the recent clustering approaches can be classified into ("flat") *partitioning* methods, and *hierarchical* methods. In this section, we will provide a brief and rather general overview of these clustering approaches together with a short list of reference methods.

2.1.1 Partitioning Algorithms

Partitioning clustering algorithms compute a "flat" partition of the data into a given number of clusters, i.e. a unique assignment of each data object to a cluster. The number of clusters k is often a user specified parameter. There are several types of partitioning methods. We review four classes of algorithms being probably the most significant ones in the field of KDD.

Optimization Based Methods try to optimize a specific clustering quality function, e.g. the average distance of the data objects in each cluster to their corresponding representative objects. This requires that each cluster is represented by a specific object. Methods usually differ in what kind of objects are used to represent cluster, e.g. objects that are part of the database, such as the medoids of the clusters, or objects that need not be part of the database, such as the mean of the points in the cluster. Typically, partitioning algorithms start with an initial partitioning of the database into k clusters. This initial partitioning may be user-defined or randomly generated. The clustering quality function determines the quality of the clustering. The initial partitioning is iteratively optimized according to the clustering quality by changing cluster representatives and reassigning objects to the new cluster representatives. If the clustering quality does not decrease after an iteration, i.e. converges, the clustering algorithms terminate. Partitioning algorithms usually converge to local minima and thus may miss the "best" clustering in terms of cluster quality. In addition, these algorithms tend to

produce clusters of spherical shapes and are rather sensitive to noise, since all objects of the database are assigned to a cluster. A further drawback of these algorithms is the sensitivity to the input parameter k (number of clusters), because the correct value of k is usually not known beforehand. Sample algorithms are k-means [McQ67], PAM [KR90], and CLARANS [NH94].

Distribution- (or Model-) Based Methods use a distribution-based quality function. Each object is assumed to be drawn from one of k underlying Gaussian distributions [JD88]. Model-based algorithms work rather similar to optimization based methods. Usually, objects are assigned to one of the k clusters using a maximum likelihood decision. Most of these algorithms have similar drawbacks as optimization based methods. Sample algorithms include the EM-algorithm [DLR77] and DBCLASD [XEKS98].

Graph Theory Based Methods model the data objects using a graph and search for connected components in that graph representing clusters. The nodes in the graph represent the data objects and an edge represents some information on the similarity of the according objects. Several approaches have been proposed, differing in the way the graph is generated. In [Zah71] a minimum spanning tree is used. Before computing the connected components, inconsistent edges (e.g. edges between two points having a distance significantly larger than the inter-point distances of the nearby edges) are removed. In [ESK03] the authors propose to use a shared k-nearest neighbor graph (objects share an edge if they share at least a minimum number of their k-nearest neighbors). The presented algorithm SNN finds clusters of arbitrary shape, different size, and different density. However, the computational complexity of the graph theoretic approaches is rather high compared to the other approaches.

Density-Based Methods search for regions of high point density that are separated by regions of lower point density. These algorithms usually need two input parameters, one specifying a volume and a second one specifying a minimum number of points. Using these two parameters, a density threshold

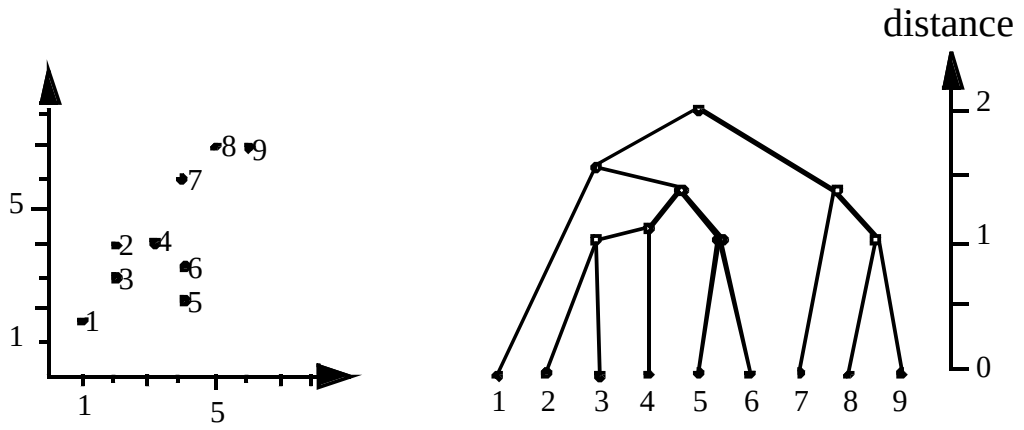


Figure 2.1: A dendrogram (right) for a sample data set (left).

is specified. Sets of objects must exceed this density threshold to be detected as clusters. The pioneering density-based approach is DBSCAN [EKSX96] which is founded on the notion of density connected sets. Since this clustering notion is the basis of this thesis, we will present the concepts underlying DBSCAN in more details in Section 2.3.

2.1.2 Hierarchical Algorithms

Hierarchical clustering algorithms compute a hierarchical decomposition of the data objects rather then a unique assignment of data objects to clusters. The hierarchical clustering structure is usually visualized by using a tree representation, a so-called *dendrogram* (cf. Figure 2.1). The leafs of a dendrogram correspond to one data object whereas the root represents the entire database. Each node in the dendrogram represents a cluster containing all clusters represented by its child nodes. Each level of the dendrogram represents a clustering of the database. An agglomerative algorithmic schema to construct a dendrogram starts with each object in the database placed in a unique cluster (leaf nodes) and then merges in each step the pair of clusters having the minimal distance until all data objects are contained in one cluster. In [Bou96] several definitions of the distance between two clusters (e.g. *single link* [Sib73]) are discussed. It is shown that each approach

yields the same result in terms of clustering quality. The combination of density-based clustering and hierarchical concepts is presented in [ABKS99]. The algorithm OPTICS is proposed to compute a density-based hierarchical decomposition of the data. In particular, OPTICS computes the clustering of DBSCAN for a broad range of parameter settings in a single scan over the data. The enhancements to density-based hierarchical clustering presented in Part II are based on the concepts of this algorithm, thus, we will discuss the concepts of OPTICS in more details in Section 2.3.

2.2 Basic Notations

Clustering relies on a notion of *similarity* among the database objects. Defining the similarity of complex data objects — such as car parts, protein molecules or text documents — is a non-trivial task, and beyond the scope of this thesis. Thus, we rely on a common approach to define similarity among the objects of a database, the so-called *feature-based* approach. The definition of similarity is often also called *similarity model*.

The key step of a feature-based similarity model is the so-called *feature extraction* or *feature transformation*. For each data object, a given number (d) of numeric features is extracted. The objects of the database are transformed into d-dimensional feature vectors, i.e. data objects are represented by points in a d-dimensional space. Then, the similarity of two data objects is measured through the proximity of the according feature vectors, e.g. using the Euclidean distance as a measure of dissimilarity. Examples of feature-based similarity include [FRM94], [MG95], and [KKM+03]. Let us note that there are other ways to define similarity on objects in a database, e.g. [Kei99], [KBK+03]. The general idea of the feature transformation of feature-based similarity is illustrated in Figure 2.2.

Throughout the rest of the thesis, we assume that $\mathcal{D}$ is a feature database of n feature vectors (or points) in some d-dimensional feature space ($\mathcal{D} \subseteq \mathbb{R}^d$). The features (or attributes) of $\mathcal{D}$ are denoted by $\mathcal{A} = \{a_1, \ldots, a_d\}$.

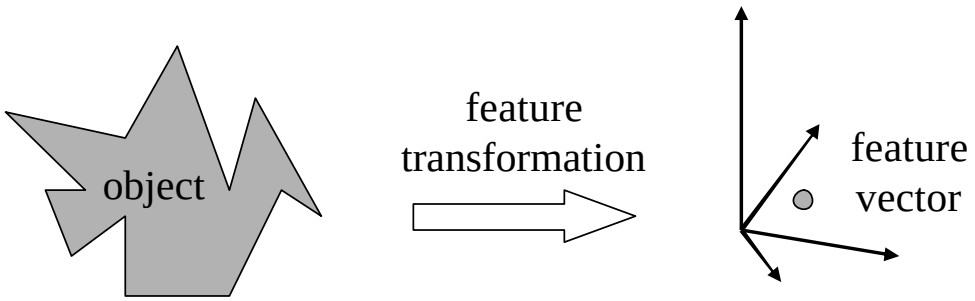


Figure 2.2: The idea of feature transformation.

A non-empty subset $S \subseteq \mathcal{A}$ is called a *subspace*. The dimensionality of a subspace $S \subseteq \mathcal{A}$, i.e. the number of attributes spanning S, is denoted by $dim[S] = |S|$. The projection of a point $p \in \mathcal{D}$ onto a subspace $S \subseteq \mathcal{A}$ is denoted by $\pi_S(p)$.

We assume that $dist : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ is a metric distance function on points in $\mathcal{D}$, indicating the dissimilarity of two objects in $\mathcal{D}$. For the sake of simplicity, we further assume $dist$ to be one of the L_p-norms which is defined for an arbitrary $p \in \mathbb{N}$ as given below:

$$dist(x, y) = \sqrt[p]{\sum_{i=1}^{d} (\pi_{\{a_i\}}(x) - \pi_{\{a_i\}}(y))^p},$$

where $x, y \in \mathcal{D}$.

Let us note that we will usually use the Euclidean distance ($p = 2$) for illustration throughout this thesis when ever it is not specified differently.

2.3 Foundations of Density-Based Clustering

The basis of density-based clustering is the observation that inside a cluster the density of points is considerably higher than outside a cluster. Furthermore, different clusters are separated by areas of noise, where the point density is lower than inside a cluster. The observation can be validated when looking at the two sample databases “database 1” and “database 2” depicted in Figure 2.3. Using the criterion of point density, we can easily and unam-

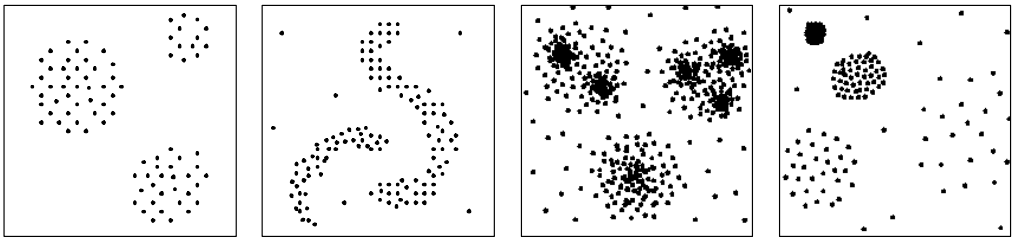


Figure 2.3: Sample databases.

biguously detect the clusters and noise points in the two sample databases. In [EKSX96], this intuitive notion of *clusters* and *noise* in a database of points in some feature space is formalized. Clusters are defined as "flat" *density connected sets*.

However, if we look at "database 3" and "database 4" depicted in Figure 2.3, we face the problem that different clusters my exhibit varying density. In addition, we may have nested clusters (cf. "database 3"), i.e. a less dense cluster may contain denser sub-clusters. To discover such hierarchical clusters, the application of hierarchical concepts is necessary. In [ABKS99] the density-based notion is extended by hierarchical concepts. Since these concepts of flat and hierarchical density-based clustering are the basis for our work, we will introduce them in detail in the following.

2.3.1 Clusters as Density Connected Sets

The key idea of "flat" density-based clustering is that for each point of a cluster the neighborhood of a given radius has to contain a minimum number of points, i.e. the density in the neighborhood has to exceed a density threshold. This threshold is determined by two user defined input parameters ε (specifying the size of the neighborhood) and *MinPts* specifying the minimum number of points the neighborhood must contain.

Definition 2.1 (ε-neighborhood)
Let $\varepsilon \in \mathbb{R}$. The ε-neighborhood *of a point $p \in \mathcal{D}$, denoted by $\mathcal{N}_\varepsilon(p)$, is*

defined by

$$\mathcal{N}_\varepsilon(p) = \{o \in \mathcal{D} \mid dist(p, o) \leq \varepsilon\}.$$

As claimed above, a point should be inside a cluster if its neighborhood contains at least a given number of points.

Definition 2.2 (core point)
A point $q \in \mathcal{D}$ is a core point *w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$, denoted by $\textsc{Core}_{den}(q)$, if its ε-neighborhood contains at least $MinPts$ points, formally:*

$$\textsc{Core}_{den}(q) \Leftrightarrow |\mathcal{N}_\varepsilon(p)| \geq MinPts.$$

Let us note, that the acronym *den* in the definition refers to 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. The core point concept is visualized in Figure 2.4(a).

A naive approach could require the core point property for each member of a cluster. However, this approach fails because there are some points on the border of the cluster (*border points*) that do not fit the core point property but are intuitively part of a cluster. In fact, a cluster has two properties: density and connectivity. The first one is captured through the core point property. The second one is captured through the following concepts.

Definition 2.3 (direct density reachable)
A point $p \in \mathcal{D}$ is direct density reachable *w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$ from $q \in \mathcal{D}$, denoted by $\textsc{DirReach}_{den}(q,p)$, if q is a core point and p is in the ε-neighborhood of q, formally:*

$$\textsc{DirReach}_{den}(q, p) \Leftrightarrow \textsc{Core}_{den}(q) \ \wedge \ p \in \mathcal{N}_\varepsilon(q).$$

The concept of direct density reachability is depicted in Figure 2.4(b). Obviously, directly density reachable is a symmetric relation for pairs of core points. However, it is not symmetric in general.

Definition 2.4 (density reachable)
A point $p \in \mathcal{D}$ *is* density-reachable *from* $q \in \mathcal{D}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and MinPts* $\in \mathbb{N}$*, denoted by* $\text{REACH}_{den}(q,p)$*, if there is a chain of points* $p_1, \ldots, p_n \in \mathcal{D}$*,* $p_1 = q$*,* $p_n = p$ *such that* p_{i+1} *is directly density reachable from* p_i*, formally:*

$$\begin{aligned}&\text{REACH}_{den}(q,p) \Leftrightarrow \\ &\quad \exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = q \ \wedge \ p_n = p \ \wedge \\ &\quad \forall i \in \{1, \ldots, n-1\} : \text{DIRREACH}_{den}(p_i, p_{i+1}).\end{aligned}$$

Density reachability is illustrated in Figure 2.4(c). It is the transitive enclosure of direct density reachable but it is not symmetric in general (again only for pairs of core points). Thus, we have captured the connectivity of core points so far. But two border points of the same cluster $\mathcal{C}$ are not density reachable from each other. However, there must be a core point in $\mathcal{C}$ from which both border points are reachable. Therefore, the following definition captures general connectivity of points within a cluster.

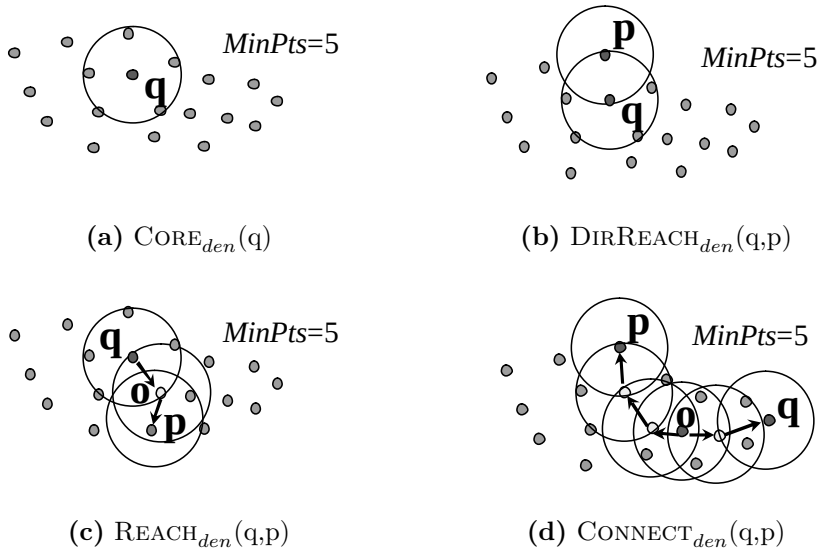
Definition 2.5 (density connected)
A point $q \in \mathcal{D}$ *is* density-connected *to another point* $p \in \mathcal{D}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and MinPts* $\in \mathbb{N}$*, denoted by* $\text{CONNECT}_{den}(q,p)$*, if there is an object* $o \in \mathcal{D}$ *such that both* p *and* q *are density reachable from* o*, formally:*

$$\begin{aligned}&\text{CONNECT}_{den}(q,p) \Leftrightarrow \\ &\exists o \in \mathcal{D} : \text{REACH}_{den}(o,q) \ \wedge \ \text{REACH}_{den}(o,p).\end{aligned}$$

Density connected is in general a symmetric relation. The concept is visualized in Figure 2.4(d).

Now, the density-based notion of a cluster can be defined using the introduced concepts. Intuitively, a cluster is defined to be a set of density connected points which is maximal w.r.t. density reachability. The points in $\mathcal{D}$ not belonging to any of its density connected sets are defined as *noise*.

Definition 2.6 (density connected set)
A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ *is called a* density connected set *w.r.t.* $\varepsilon \in \mathbb{R}$



(a) $\textsc{Core}_{den}$(q) **(b)** $\textsc{DirReach}_{den}$(q,p)

(c) $\textsc{Reach}_{den}$(q,p) **(d)** $\textsc{Connect}_{den}$(q,p)

Figure 2.4: Illustration of density-based clustering concepts

and $MinPts \in \mathbb{N}$*, if all objects in* $\mathcal{C}$ *are density-connected, formally:*

$$\textsc{ConSet}_{den}(\mathcal{C}) \Leftrightarrow \forall p, q \in \mathcal{C} : \textsc{Connect}_{den}(p, q)$$

Definition 2.7 (density connected cluster)
A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ *is called a* density connected cluster *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$*, denoted by* $\textsc{Cluster}_{den}(\mathcal{C})$*, if* $\mathcal{C}$ *is a density connected set and* $\mathcal{C}$ *is maximal w.r.t. density-reachability, formally:*

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

(1) *Connectivity:* $\textsc{ConSet}_{den}(\mathcal{C})$

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

We will use the terms “density-based” and “density connected” throughout the rest of the thesis interchangeable for the clustering notion as defined in Definition 2.7. Note, that the density connected clustering notion is able to detect clusters of arbitrary shape and size as long as they exceed the threshold. A flat density-based decomposition of a database is defined as follows.

```
algorithm DBSCAN(SetOfObjects 𝒟, Real ε, Integer MinPts)
  // each point in 𝒟 is marked as unclassified
  generate new clusterID cid;
  for each p ∈ 𝒟 do
    if p.clusterID = UNCLASSIFIED then
      if ExpandCluster(𝒟, p, cid, ε, MinPts) then
        cid := cid + 1
      end if
    end if
  end for
```

Figure 2.5: The DBSCAN algorithm.

Definition 2.8 (flat density-based decomposition)
Let $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$. A flat density-based decomposition *of $\mathcal{D}$ w.r.t. ε and $MinPts$ is a decomposition D_{den} of $\mathcal{D}$ into $k \geq 1$ subsets, such that $k-1$ subsets are density connected clusters and the k-th (possible empty) set contains the noise points, formally:*

$$D_{den} = \{C_1, \ldots, C_{k-1}, N\} \qquad \textit{where}$$
$$\neg \textsc{Cluster}_{den}(N) \wedge \forall i : i \in \{1, \ldots, k-1\} \wedge C_i \neq \emptyset \wedge \textsc{Cluster}_{den}(C_i).$$

Using the previously described concepts, the algorithm DBSCAN is proposed in [EKSX96] computing a flat density-based decomposition w.r.t. the user-specified parameters ε and *MinPts* by one single pass over the data. For that purpose, DBSCAN uses the fact, that a density connected set can be detected by finding one of its core points p and computing all objects which are density reachable from p. The pseudo code of DBSCAN is depicted in Figure 2.5. The method ExpandCluster which computes the density connected cluster starting from a given core point, is given in Figure 2.6.

The correctness of DBSCAN can be formally proven (cf. Lemmata 1 and 2 in [EKSX96], proofs in [SEKX98]). 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 spatial index (e.g. [BKK96] or [BBJ+00]) and $O(n^2)$ if no index exists.

```
boolean ExpandCluster(SetOfObjects 𝒟, Object start, Integer cid, Real ε, Integer MinPts)
  SetOfObjects seeds := 𝒩_ε(start);
  if |seeds| < MinPts then
    start.clusterID := NOISE;
    return false;
  end if
  for each o ∈ seeds do
    o.clusterID := cid;
  end for
  remove start from seeds;
  while seeds ≠ ∅ do
    o := first point in seeds;
    neighbors := 𝒩_ε(o);
    if |neighbors| ≥ MinPts then
      for each p ∈ neighbors do
        if p.clusterID ∈ {UNCLASSIFIED, NOISE} then
          if p.clusterID = UNCLASSIFIED then
            insert p into seeds;
          endif
        p.clusterID := cid;
        endif
      end for
    end if
    remove o from seeds;
  end while
  return true;
```

Figure 2.6: Method ExpandCluster.

2.3.2 Density-Based Hierarchical Decompositions

DBSCAN computes a flat density-based decomposition of a database. It detects each density connected set w.r.t. a global density parameter specified by ε and *MinPts*. However, there may be clusters of different density and/or nested clusters in the database (cf. "database 3" and "database 4" in Figure 2.3). If the densities of different clusters vary significantly, the parameterization of DBSCAN gets problematic. A less strict density threshold would detect also the clusters of lower density but may merge clusters of higher density. On the other hand, a more strict density threshold would partition the denser clusters but would miss clusters with lower density. In addition, the information of nested clusters, i.e. denser clusters within less dense clusters, may be missed.

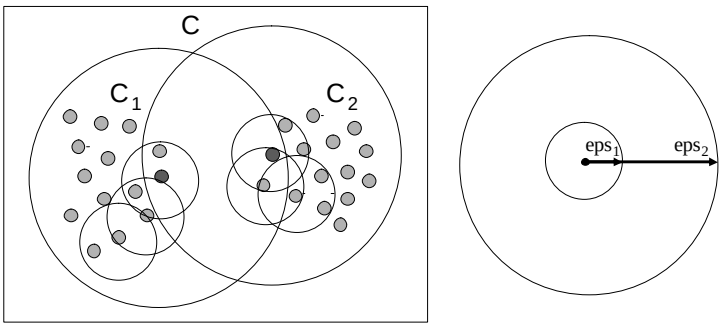


Figure 2.7: Nested clusters of different density.

In [ABKS99], the density connected clustering notion is extended by hierarchical concepts. Based on these concepts, the algorithm OPTICS is presented. The key idea is, that (for a constant *MinPts*-value) density-based clusters w.r.t. a higher density (i.e. a lower value for ε) are completely contained in density-based clusters w.r.t. a lower density (i.e. a higher value for ε). Figure 2.7 illustrates this observation: C_1 and C_2 are density-based clusters w.r.t. $eps_1 < eps_2$ and C is a density-based cluster w.r.t. eps_2 completely containing C_1 and C_2.

The algorithm OPTICS works like an extended DBSCAN algorithm, computing the density connected clusters w.r.t. all parameters ε_i that are smaller than a generic value ε. In contrast to DBSCAN, OPTICS does not assign cluster memberships, but stores the order in which the data objects are processed and the information which would be used by an extended DBSCAN algorithm to assign cluster memberships. This information consists of only two values for each object, the *core distance* and the *reachability distance*.

The core distance is based on the concept of k-nearest neighbor distances.

Definition 2.9 (k-nearest neighbor distance)
Let $k \in \mathbb{N}$. The k-nearest neighbors *of a point $p \in \mathcal{D}$ is the smallest set $NN_k(p) \subseteq \mathcal{D}$ that contains (at least) k points from the database, and for which the following condition holds:*

$$\forall o \in NN_k(p), \forall q \in \mathcal{D} - NN_k(p) : dist(o,p) < dist(q,p).$$

The k-nearest neighbor distance *of p, denoted by $nn\text{-}dist_k(p)$ is defined as*

follows:

$$\textit{nn-dist}_k(p) = \max\{dist(o,p) \mid o \in NN_k(p)\}.$$

Let us note that in Definition 2.9 it is implicitly assumed that $\mathcal{D}$ contains at least k elements, i.e. $k \leq n$.

Definition 2.10 (core distance)
The core distance *of a point $q \in \mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and $\mathit{MinPts} \in \mathbb{N}$ is defined as*

$$\textsc{CoreDist}_{den}(q) = \begin{cases} \textit{nn-dist}_{\mathit{MinPts}}(q) & \text{if } |\mathcal{N}_\varepsilon(q)| \geq \mathit{MinPts} \\ \infty & \text{else.} \end{cases}$$

The core distance of a point q is the smallest threshold $\hat{\varepsilon} \leq \varepsilon$ such that q is a core point w.r.t. $\hat{\varepsilon}$ and *MinPts*. If $\hat{\varepsilon}$ would be greater than the generic ε value, the core distance of q is set to ∞.

Definition 2.11 (reachability distance)
The reachability distance *of a point $p \in \mathcal{D}$ relative from another point $q \in \mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and $\mathit{MinPts} \in \mathbb{N}$ is defined as*

$$\textsc{ReachDist}_{den}(q,p) = \max(\textsc{CoreDist}_{den}(q), dist(q,p)).$$

The reachability distance of a point p w.r.t. another point q is the smallest threshold $\hat{\varepsilon} \leq \varepsilon$ such that p is directly density reachable from q. Obviously, to achieve this relation, q has to be a core point. Thus, the reachability distance cannot be smaller than the core distance of q. As a consequence, if $dist(q,p) \leq \textsc{CoreDist}_{den}(q)$, the reachability distance of p w.r.t. q is set to $\textsc{CoreDist}_{den}$(q). Otherwise, the smallest threshold $\hat{\varepsilon} \leq \varepsilon$, such that p is directly density reachable from q, is exactly $dist(q,p)$. Let us note that if q is not a core point w.r.t. the generic ε-value (i.e. $\textsc{CoreDist}_{den}(q) = \infty$), we get $\textsc{ReachDist}_{den}(q,p) = \infty$ indicating that the smallest threshold $\hat{\varepsilon}$ is in fact greater than ε, i.e. p cannot be directly reached from q w.r.t. the generic threshold ε.

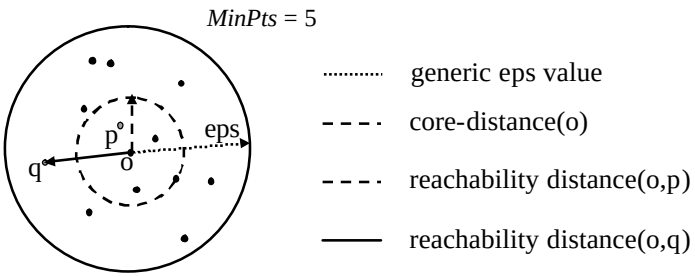


Figure 2.8: Illustration of core distance and reachability distance.

Both the core distance of a point o and the reachability distances of the points p and q relative to o are illustrated in Figure 2.8.

The OPTICS algorithm computes a so-called *cluster ordering* of a database w.r.t. the two input parameters ε and *MinPts*. In addition, the core distance and a "suitable" reachability distance is stored for each object. The pseudo code of the OPTICS algorithm is depicted in Figure 2.9. It starts with an arbitrary point $o \in \mathcal{D}$, assigns a reachability distance of ∞ to o and expands the cluster order if the core distance of o is smaller than the generic (input parameter) ε. The expansion is worked out by inserting each point $p \in \mathcal{N}_\varepsilon(o)$ into a seed list OrderedSeeds. This seed list is organized as a heap, storing that point q as first object in the list, having the minimum reachability distance to the already processed points. The heap structure is maintained by the procedure OrderedSeeds::update (cf. Figure 2.10) which updates the reachability distances of the points that are already in the seed list if their according values decrease. The next point to be inserted in the cluster ordering is always the first object in the seed list. If the core distance of this object is smaller or equal to ε, all points in the ε-neighborhood are again inserted into or updated in the seed list. If the seed list is empty and there are still some not yet processed points in $\mathcal{D}$, we have a so-called "jump". OPTICS selects another arbitrary not yet handled point in $\mathcal{D}$ to further expand the cluster ordering CO as described above.

Definition 2.12 (cluster ordering)
Let MinPts $\in \mathbb{N}$, $\varepsilon \in \mathbb{R}$, *and* CO *be a permutation of the objects in* $\mathcal{D}$. *Each* $o \in \mathcal{D}$ *has additional attributes* $o.P$, $o.C$ *and* $o.R$, *where* $o.P \in \{1, \ldots, n\}$

```
algorithm OPTICS(SetOfObjects 𝒟, Real ε, Integer MinPts)
  CO := empty cluster ordering;
  while |CO| < n do
    o := arbitrary not yet handled point in 𝒟;
    neighbors_o := 𝒩_ε(o);
    o.R := ∞;
    o.C := Core_den(o);
    mark o as handled;
    append o to CO;
    if o.C ≠ ∞ then
      OrderedSeeds.update(neighbors_o, o);
      while OrderedSeeds ≠ ∅ do
        p := OrderedSeeds.first();
        neighbors_p := 𝒩_ε(p);
        p.C := Core_den(p);
        mark p as handled;
        append p to CO;
        if p.C ≠ ∞ then
          OrderedSeeds.update(neighbors_p, p);
        end if
      end while
    end if
  end while
```

Figure 2.9: The OPTICS algorithm.

symbolizes the position of o in CO. We call CO a cluster ordering *w.r.t. ε and MinPts if the following three conditions hold:*

(1) $\forall p \in CO : p.C = \text{CoreDist}_{den}(p)$

(2) $\forall x, y \in CO : 1 < x.P < y.P \Rightarrow$

$$\forall o \in CO : o.P < x.P \Rightarrow \text{ReachDist}_{den}(o, x) \leq \text{Reach}_{den}(o, y)$$

(3) $\forall p \in CO :$

$$p.R = \min\{\text{ReachDist}_{den}(o, p) \mid o \in CO \wedge o.P < p.P\},$$

where $\min \emptyset = \infty$.

Intuitively, condition (2) states that the order is built on selecting at each position i in CO that object o having the minimum reachability to any object before i. $o.C$ symbolizes the core distance of an object o in CO whereas $o.R$ is the reachability distance assigned to object o during the generation of CO.

```
method OrderedSeeds::update(SetOfObjects neighbors, Object center)
  cdist := center.C;
  for each o ∈ neighbors do
    if o is not yet processed then
      rdist := max{cdist, dist(o, center)};
      if o is already in OrderedSeeds then
        if o.R > rdist then
          o.R := rdist;
          decrease(o);
        end if
      else
        o.R := rdist;
        insert(o);
      end if
    end if
  end for
```

Figure 2.10: Method OrderedSeeds::update.

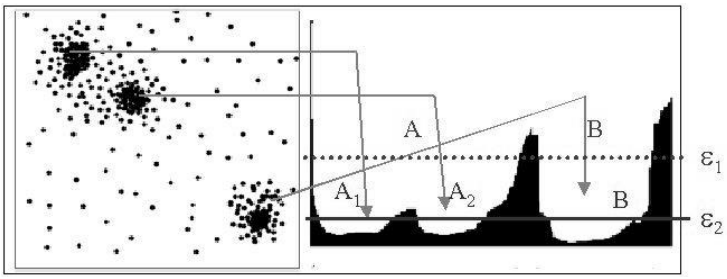


Figure 2.11: Reachability plot (right) computed by OPTICS for a sample 2-D data set (left).

A cluster ordering contains sufficient information to extract all density-based clusterings w.r.t. any $\varepsilon' \leq \varepsilon$. The density-based clustering w.r.t. a particular $\varepsilon' \leq \varepsilon$ can be extracted by scanning the cluster ordering and checking the reachability distance and the core distance of each object. If the reachability distance of the current object is larger than ε', we have to check its core distance. If the core distance of this object is also larger than ε', this object is assigned to noise. Else, the object is a core object and we start a new cluster. If the reachability of the current object is smaller than ε', it can be assigned to the current cluster because it is density reachable from a preceding core point in the cluster ordering. Let us note, that the resulting clusters may miss some border points.

A breakthrough advantage of OPTICS is that the resulting cluster ordering can be visualized very intuitively and clearly by means of a so-called *reachability plot*. A reachability plot is a 2-dimensional visualization of a cluster ordering, where the points are plotted according to the sequence specified in the cluster ordering along the x-axis, and for each point, the reachability distance along the y-axis. Figure 2.11 (right) depicts the reachability plot based on the cluster ordering computed by OPTICS for the sample 2-dimensional data set in Figure 2.11 (left). Intuitively, clusters are "valleys" or "dents" in the plot, because sets of consecutive points with a lower reachability value are packed more densely. In particular, to manually obtain a density-based clustering w.r.t. any $\varepsilon' \leq \varepsilon$ by visual analysis, one simply has to cut the reachability plot at y-level ε' (i.e. parallel to the x-axis). The consecutive valleys in the plot below this cutting line contain the according clusters. An example is presented in Figure 2.11: For a cut at the level ε_1, we find two clusters denoted as A and B. Compared to this clustering, a cut at level ε_2 would yield three clusters. The cluster A is split into two smaller clusters denoted by A_1 and A_2 and cluster B decreased its size. This illustrates how the hierarchical cluster structure of a database is revealed at a glance and could be easily explored by visual inspection.

Part II

Using Density-Based Hierarchical Clustering for Similarity Search Applications

Chapter 3

A Browsing Tool for Similarity Search

In the last ten years, an increasing number of database applications has emerged for which efficient and effective support for similarity search is substantial. In this chapter, we outline the application of OPTICS to similarity search. First, we sketch the use and the benefit of density-based hierarchical clustering for different application ranges related to similarity search and visual data mining in Section 3.1, including visual data mining, similarity search, and evaluation of similarity models. These considerations motivate and initiate the development of an interactive data browsing tool called BOSS (Browsing OPTICS plots for Similarity Search). Second, we identify two key requirements necessary for the realization of the BOSS prototype in Section 3.2 which represent novel challenges for density-based clustering. The challenges will be addressed in the consecutive chapters of this part of the thesis.

3.1 Motivation

The importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer aided engineering, marketing and purchasing assistance, etc. (e.g. [Jag91], [AFS93], [MG95], [FRM94], [ALSS95], [BK97], [Kei99]). Particularly, the task of finding similar shapes in 2-dimensional and 3-dimensional spaces becomes more and more important. Examples for new applications that require the retrieval of similar 3-dimensional objects include databases for molecular biology, medical imaging and computer aided design.

In this section, we outline novel application ranges of density-based hierarchical clustering which led to the development of an interactive browsing tool, called BOSS (Browsing OPTICS plots for Similarity Search). The core idea of BOSS is to provide a browsable hierarchy of clusters each represented by one or more significant objects. We will describe the details of BOSS and evaluate the prototype in Chapter 6. To generate a cluster hierarchy, BOSS makes use of the density-based hierarchical clustering algorithm OPTICS which was introduced in Section 2.3. The use of OPTICS yields several advantages due to the following reasons:

- OPTICS is — in contrast to most other algorithms — relatively insensitive to its two input parameters, ε and *MinPts*. The authors in [ABKS99] state that the input parameters just have to be large enough to produce good results.
- OPTICS is a hierarchical clustering method which yields more information about the cluster structure than a method that computes a flat partitioning of the data (e.g. DBSCAN [EKSX96]).
- OPTICS is applicable and scalable to large databases. The performance of OPTICS can be significantly improved through speeding-up the range queries, using appropriate spatial index structures (e.g. [BBJ+00], [CPZ97]).
- The result of OPTICS is a cluster ordering. Using reachability plots,

cluster orderings can be visualized much more clear than dendrograms especially for large data sets.

BOSS was designed for three different purposes: visual data mining, similarity search and the evaluation of similarity models. For the first two applications, the choice of the representative objects of a cluster is the key step. It helps the user to get a meaningful and quick overview of a large existing data set. Furthermore, BOSS helps scientists to evaluate new similarity models.

3.1.1 Visual Data Mining

As defined in [Ank00], visual data mining is a step in the KDD process that utilizes visualization as a communication channel between the computer and the user to produce novel and interpretable patterns. Based on the balance and sequence of the automatic and the interactive (visual) part of the KDD process, three classes of visual data mining can be identified.

- **Visualization of the data mining result:** An algorithm extracts patterns from the data. These patterns are visualized to make them interpretable. Based on the visualization, the user may want to return to the data mining algorithm and run it again with different input parameters (cf. Figure 3.1a).
- **Visualization of an intermediate result:** An algorithm performs an analysis of the data not producing the final patterns but an intermediate result which can be visualized. Then the user retrieves the interesting patterns in the visualization of the intermediate result (cf. Figure 3.1b).
- **Visualization of the data:** Data is visualized immediately without running a sophisticated algorithm before. Patterns are obtained by the user by exploring the visualized data (cf. Figure 3.1c).

The approach presented in this thesis belongs to the first and second class. A hierarchical clustering algorithm is applied to the data which extracts the

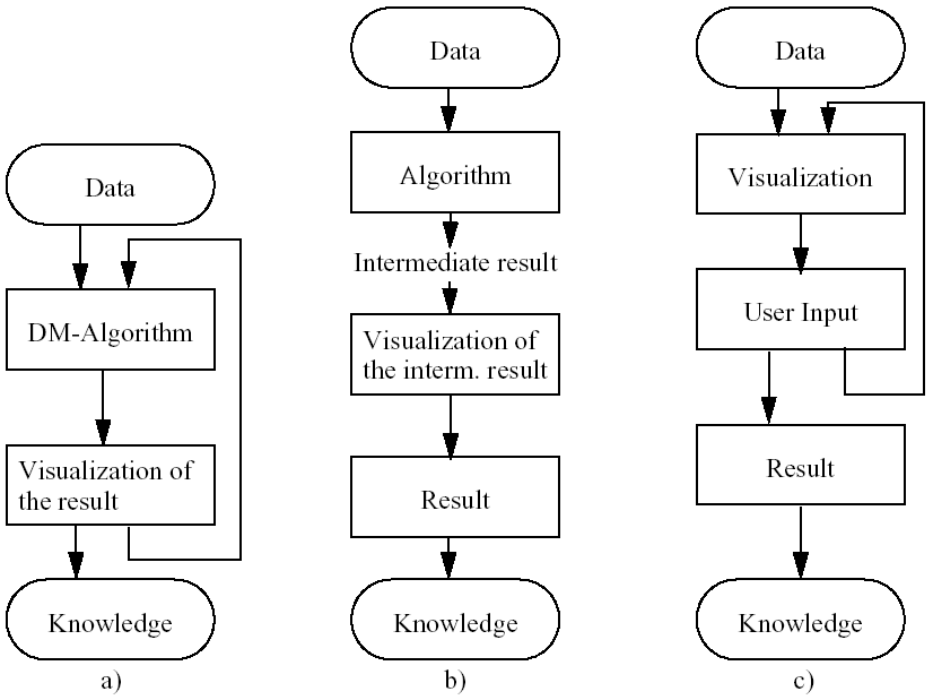


Figure 3.1: Different approaches to visual data mining [Ank00].

clustering structure as an intermediate result. There is no meaning associated with the generated clusters. However, our approach allows the user to visually analyze the contents of the clusters. The clustering algorithm used in the algorithmic part is independent from an application. It performs the core part of the data mining process and its result serves as a multi-purpose basis for further analysis directed by the user. This way, the user may obtain novel information which was not even known to exist in the data set. This is in contrast to similarity search where the user is restricted to find similar parts respective to a query object and a predetermined similarity measure.

3.1.2 Similarity Search

The development, design, manufacturing and maintenance of modern engineering products is a very expensive and complex task. Effective similarity

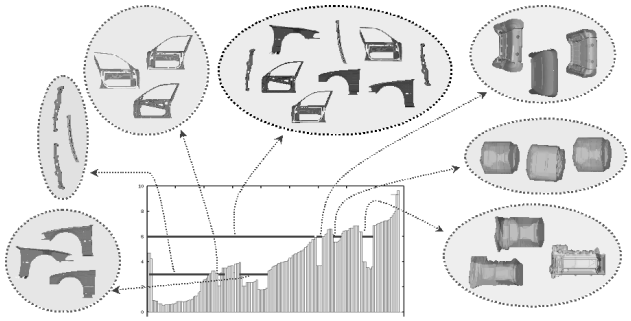


Figure 3.2: Browsing through reachability plots.

models are required for two- and three-dimensional CAD applications to cope with rapidly growing amounts of data. Shorter product cycles and a greater diversity of models are becoming decisive competitive factors in the hard-fought automobile and aircraft market. These demands can only be met if the engineers have an overview of already existing CAD parts. It would be desirable to have an interactive data browsing tool which depicts the reachability plot computed by OPTICS in a user friendly way together with appropriate representatives of the clusters. This clear illustration would support the user in his time-consuming task to find similar parts. It is rather obvious, that other application domains to similarity search beside CAD, e.g. molecular biology and multimedia, would also benefit from such a tool. From the user's point of view, this browsing tool should meet the following two requirements:

- The hierarchical clustering structure of the data set is revealed at a glance. The reachability plot is an intuitive visualization of the cluster hierarchy which helps to assign each object to its corresponding cluster or to noise. Furthermore, the hierarchical representation of the clusters using the reachability plot helps the user to get a quick overview of all clusters and their relation to each other. As each entry in the reachability plot is assigned to one object, we can easily illustrate some representatives of the clusters belonging to a given density threshold (cf. Figure 3.2).

- The user is not only interested in the shape and the number of the

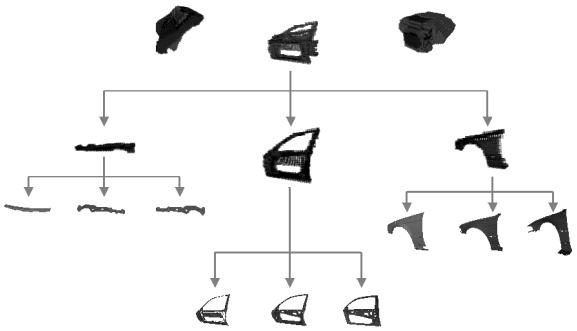


Figure 3.3: Hierarchically ordered representatives.

clusters, but also in the specific objects building up a cluster. As for large clusters it is rather difficult to depict all objects, suitable representatives of each cluster should be displayed. To follow up a first idea, these representatives could be simply constructed by superimposing all parts belonging to the regarded cluster (cf. Figure 3.3). We can browse through the hierarchy of the representatives in the same way as through the OPTICS plots.

This way, the cost of developing and producing new parts could be reduced by maximizing the reuse of existing parts, because the user can browse through the hierarchical structure of the clusters in a top-down way. Thus, the engineers get an overview of already existing parts and are able to navigate their way through the diversity of existing variants of products, such as cars.

3.1.3 Evaluation of Similarity Models

In general, similarity models can be evaluated by computing k-nearest neighbor queries (k-nn queries). However, this evaluation approach is subjective and error-prone because the quality measure of the similarity model depends on the results of a few similarity queries and, therefore, on the choice of the query objects. A model may perfectly reflect the intuitive similarity according to the chosen query objects and would be evaluated as "good" although it produces disastrous results for the vast majority of database objects. On

the other hand, one may choose a database object which is rather unique, i.e. there are no similar objects to the query object in the database (in terms of clustering, we would speak of a noise object). Based on the results for such a query object, a model will be evaluated as bad, unless it may represent the intuitive notion of similarity for the data objects quite well. In addition, the parameter k of the k-nn queries may not be suitable for all query objects. There may be objects that have only very few similar objects in the database (e.g. only two wings in a data set of aircraft parts) whereas other objects may have many similar objects (e.g. hundreds or thousands of screws in the same data set).

A better way to evaluate and compare several similarity models is to apply a clustering algorithm [KKM+03]. It is more objective since each object of the data set is taken into account to evaluate the data models rather than some sample objects. In fact, the results of a hierarchical clustering algorithm such as OPTICS is rather suitable to evaluate and compare several similarity models. Our browsing tool BOSS extremely simplifies the comparisons of two or more reachability plots generated by OPTICS applied to different similarity models.

3.2 Required Enhancements

The BOSS prototype uses the information of a reachability plot generated by OPTICS to support the three sketched applications by visualizing the hierarchical clustering structure, generating a hierarchy of clusters, and revealing representative objects of each cluster. However, some steps in the pipeline from the raw data to the interactive browsing remain unsolved so far. In particular, to enable browsing a hierarchy of cluster representatives extracted from a reachability plot, several additions and enhancements to the hierarchical clustering algorithm OPTICS are necessary. In the following, we work out two requirements that were needed to be addressed during the development of BOSS.

Requirement 1: Incremental Clustering
The browsing tool BOSS is supposed to be applied to large dynamic databases, i.e. the databases contain a huge amount of objects and updates — insertion of new objects and deletion of objects — frequently occur. In such a dynamic environment, the detected clustering structure most likely changes due to update operations. For consistency reasons, the computed clusters have to be updated as well. Rerunning OPTICS applied on the updated database to rearrange the cluster ordering (and thus the reachability plot) is obviously rather inefficient. It would be much more sensible to rearrange only that parts of the cluster ordering being affected by the update (if this is even possible). In fact, incrementally updating the clustering structure of a large database is mandatory for interactive systems in a dynamic environment. Due to availability reasons, the clusters should not be recomputed from scratch after a database update operation. Since — to the best of our knowledge — there has been no incremental version of OPTICS proposed so far, this is a first urgent requirement for our prototype.

Requirement 2: Cluster Recognition and Representation
Solid cluster extraction and the determination of meaningful cluster representatives form the foundation for providing the user with significant information and a quick overview of the objects in the database and to enable interactive browsing. Thus, the extraction of clusters from a cluster ordering or reachability plot is a mandatory component of BOSS. In addition, the generation of meaningful representative objects for the recognized clusters is a key feature for the visual browsing facility. Both cluster recognition and cluster representation together are a second requirement for our prototype.

In the following, we will meet both requirements. In Chapter 4 an incremental version of the OPTICS algorithm (cf. Requirement 1) is presented to scale the ideas of BOSS to a dynamic environment. Novel solutions for cluster recognition and cluster representation from reachability plots computed by OPTICS (cf. Requirement 2) are introduced in Chapter 5. All these concepts extend the basic OPTICS algorithm to cope with novel in-

dustrial applications. Chapter 6 describes some technical details of the BOSS prototype and presents some sample applications of BOSS.

Chapter 4

Incremental Clustering

The interactive browsing tool BOSS is supposed to be applied to dynamic environments, i.e. databases in which updates frequently occur. Thus, one major challenge for density-based hierarchical clustering identified in Section 3.2 is an incremental version of the OPTICS algorithm. In this chapter, we address this requirement. First, we review and discuss related work on incrementally updating mined patterns in dynamic databases in Section 4.1. Then, we present the modifications and extensions of density-based hierarchical concepts, necessary for the development of an incremental OPTICS algorithm in Section 4.2. In addition, an incremental algorithm called IncOPTICS is proposed. The concepts proposed in this section are major extensions of the material published in [KKG03]. A broad experimental evaluation of the performance of IncOPTICS compared to the original OPTICS algorithm is presented in Section 4.3. Section 4.4 concludes this chapter with a short summary.

4.1 Related Work

The problem of incrementally updating mined patterns is a rather new area of research. Most work has been done in the area of developing incremental algorithms for the task of mining association rules [AS94]. Incremental approaches include [CHNW96] and [FAAM97]. In [EW98] algorithms for incremental attribute-oriented generalization are presented.

The problem of incremental clustering has been addressed in several publications recently. In the following, we present some of the proposed methods without the sake of completeness.

In [Sib73] the SLINK algorithm for single-linkage clustering is presented. Based on this work, in [Def77] a similar method called CLINK for complete-linkage clustering is proposed. Both algorithms work recursively, i.e. they compute a dendrogram of n data objects by starting with a dendrogram of one object and then recursively inserting the remaining $n-1$ object. The fundamental concept of both approaches is the so-called *pointer representation* which is a compact representation of a dendrogram. In both algorithms, the insert of one object into a dendrogram represented by the pointer representation of n objects has a run time complexity of $O(n)$. Using SLINK or CLINK, one can only incrementally update inserts. Due to the recursive property, deletions cannot be handled incrementally.

In [CHO02] the incremental hierarchical clustering algorithm GRIN is proposed. GRIN uses GRACE, an agglomerative hierarchical clustering algorithm, to produce a dendrogram for the data set. After that, the bottom levels of this dendrogram is pruned. Each cluster in the resulting dendrogram is represented by the centroid, the radius, and the number of points of the cluster. New data points are inserted into leaf nodes based on the gravity theory in physics.

The agglomerative incremental hierarchical clustering algorithm IHC is proposed in [WIY02]. IHC defines the homogeneity of clusters and monotonicity of the cluster hierarchy. New points are inserted bottom-up into the hierarchy. A reconstruction procedure repairs clusters whose homogeneity

has decreased by eliminating lower and higher dense regions.

In [EKS+98] an incremental version of DBSCAN called IncrementalDBSCAN is proposed. Due to its density-based clustering notion (cf. Section 2.3), IncrementalDBSCAN limits the effects of an update operation only to the neighborhood of the update object. In fact, IncrementalDBSCAN yields tremendous speed-up factors even for a huge bulk of updates compared to rerunning DBSCAN from scratch.

Let us note that there are several incremental solutions to the clustering of stream data (e.g. [Bar02], [AHWY03], [OMM+02], [GGR02]) which is a related problem. Usually these methods pay off accuracy for efficiency reasons.

In [BKKS01] a compression technique for hierarchical clustering called "Data Bubbles" is proposed, yielding a huge speed-up for clustering of vector data by paying only a small decrease of clustering quality. Recently, this approach was extended for non-vector data (i.e. arbitrary metric data). The extended version outperforms the original Data Bubbles in terms of clustering quality on vector data [ZS03]. In [NSC04], the authors propose an incremental summarization method based on these Data Bubbles which can be applied to dynamic hierarchical clustering. In particular, Data Bubbles are tested using OPTICS, providing a suitable possibility to speed-up the construction and incrementally maintain a cluster ordering computed by OPTICS. However, Data Bubbles are not applicable to BOSS because, due to the data summarization, they tend to miss details in the cluster hierarchy by increasing the compression rate. As a consequence, important details in the cluster hierarchy cannot be browsed by BOSS, because the details are simply not present in the cluster ordering.

4.2 Incremental OPTICS

Our ideas of BOSS rely on the use of OPTICS as a clustering algorithm (cf. Section 3.1), thus, recent approaches for incrementally maintaining a flat or

hierarchical clustering structure cannot be applied. We need an incremental version of OPTICS which will be presented in the following.

4.2.1 General Ideas and Concepts

The key idea of an incremental version of OPTICS is to reconstruct only parts of the cluster ordering that are affected by the update operation rather than recomputing the entire cluster ordering from scratch. Recall from Section 2.3 that the cluster ordering is generated by starting from an arbitrary point in $\mathcal{D}$ and then at each position i in the cluster ordering, selecting that point having the smallest reachability distance to the already processed points (i.e. points coming before i in the cluster ordering). In case of an update operation, this order may be violated. In particular, conditions (2) and (3) of Definition 2.12 may be violated such that the cluster ordering is not valid anymore.

Theorem 4.1
Let CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$. An update (insert/delete) operation may affect the entire cluster ordering.

Proof. *By example: Consider a 2-dimensional data set containing 2 clusters and an additional point A — cluster 1 contains point B and cluster 2 contains points C and D. Let us assume that OPTICS starts with point A which does not belong to any cluster. At next, assume that point B has the smallest reachability value w.r.t. point A, directing the run of OPTICS into cluster 1. After all points in cluster 1 are worked out, let us assume that point D from cluster 2 is the next point in the seed list with a minimal reachability distance w.r.t. the already processed points. Thus, OPTICS will enter cluster 2 via point D. Let us further assume that point C is visited as last point.*

An insertion of a point U near A and C may affect the run of OPTICS in the following way: from point A we now may visit points U and C instead of point B. Thus, OPTICS will now run through cluster 2 first, and after

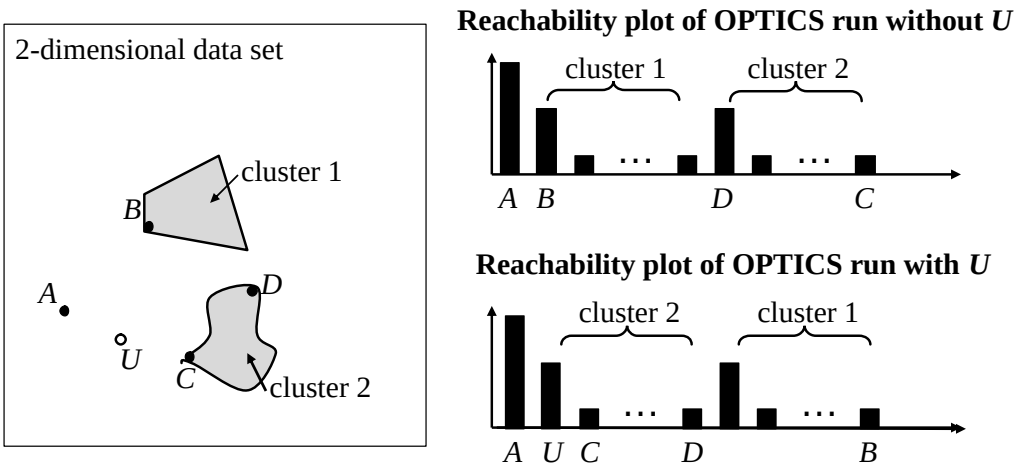


Figure 4.1: Sample dataset where the entire cluster ordering is affected by the insertion/deletion of point U.

that through cluster 1 to stop at B. Obviously, we processed the points after the insertion in reverse order.

Deletion analogously. □

The example constructed in the proof is illustrated in Figure 4.1

The theorem states that it cannot be guaranteed that a complete recomputation of the ordering is never needed. The path OPTICS chooses through the database can be completely different after an update because reachability connections change. For each changed connection, we need the computation of one range query. A range query can be computed in $O(n)$ time if not using an index, and can be accelerated to $O(\log n)$ using a spatial index structure such as the X-Tree [BKK96] or the IQ-Tree [BBJ+00]. However, in most cases, several connections remain unchanged. The idea of an incremental update is to save as much of the range queries, necessary for a rerun, as possible during an incremental reconstruction. In the following, we say that if a point participates in a change of connections, this point is affected by the update and need a reorganization.

To determine the objects that are affected by an update operation, i.e. need reorganization to re-establish a valid cluster ordering, we make the following considerations: Due to an update (insert/delete) operation, the core distance of some points may change. As a consequence, the reachability distances of some objects that were "reached" from these points in the cluster ordering may also change, causing the above mentioned violation of condition (2) in Definition 2.12. Thus, in a first step, it is important to determine the points with changing core levels, and then, in a second step, to determine the objects that are affected by these changes. The following considerations are based on the concepts described in Section 2.3.

Definition 4.1 (reverse k-nearest neighbors)
Let $k \in \mathbb{N}$. The reverse k-nearest neighbors *of a point $p \in \mathcal{D}$, denoted by $\textsc{Rev}_k(p)$, is defined as*

$$\textsc{Rev}_k(p) = \{q \in \mathcal{D} \mid p \in NN_k(q)\}.$$

Let us note that we implicitly assume that $\mathcal{D}$ contains at least k objects, i.e. $k \leq n$.

Based on the concepts of reverse k-nearest neighbors, we can identify the objects changing their core distance in a given cluster ordering.

Definition 4.2 (points with changing core distances)
Let p be a point either in or not yet in $\mathcal{D}$ and CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$. The set of points with changing core distances *due to insertion/deletion of a point $p \in \mathcal{D}$, denoted by $\textsc{Change}_{den}(p)$, is defined as*

$$\textsc{Change}_{den}(p) = \{q \in \textsc{Rev}_{MinPts}(p) \mid dist(q,p) \leq \varepsilon\}$$

Let us note that we use the acronym *den* for the density parameter ε and *MinPts* wherever they are clear from context.

In fact, not all points in $\textsc{Change}_{den}(p)$ must change their core distances. It may happen that a point in $\textsc{Change}_{den}(p)$ has a core distance of already

∞. In case of deleting p, the core distance remains unchanged (the *MinPts*-nearest neighbor distance further grows). In case of inserting p, the change of the core distance depends on whether the *MinPts*-nearest neighbor distance decreases under the limit of ε. If so, the core distance of the point changes, otherwise not. However, all points not belonging to $\text{CHANGE}_{den}(p)$ cannot change its core distance due to the insertion/deletion of p.

The set $\text{CHANGE}_{den}(p)$ can be efficiently computed using an index structure for reverse nearest neighbor queries such as proposed in [KM00] or [YL01]. However, both approaches suffer from high update costs (for the according index structure) and are only proposed for the support of reverse 1-nearest neighbor queries. Nevertheless, $\text{CHANGE}_{den}(p)$ for an update point p can be computed rather efficiently, due to the following lemma.

Lemma 4.1

Let p be a point either in or not yet in $\mathcal{D}$ and CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$. Then $\text{CHANGE}_{den}(p)$ can be computed as follows:

$$\text{CHANGE}_{den}(p) = \{q \mid q \in \mathcal{N}_\varepsilon(p) \wedge dist(p,q) \leq \text{CORE}_{den}(q)\}.$$

Proof.

Let $X := \{q \mid q \in \mathcal{N}_\varepsilon(p) \wedge dist(p,q) \leq \text{CORE}_{den}(q)\}$.

We show that $\text{CHANGE}_{den}(p) = X$*:*

$\forall q \in \text{CHANGE}_{den}(p)$

$\overset{\text{Def 4.2}}{\Longleftrightarrow} dist(q,p) \leq \varepsilon \wedge q \in \text{REV}_{MinPts}(p)$

$\overset{\text{Def 2.1}}{\Longleftrightarrow} q \in \mathcal{N}_\varepsilon(p) \wedge q \in \text{REV}_{MinPts}(p)$

$\overset{\text{Def 4.1}}{\Longleftrightarrow} q \in \mathcal{N}_\varepsilon(p) \wedge p \in NN_{MinPts}(q)$

$\overset{\text{Def 2.9}}{\Longleftrightarrow} q \in \mathcal{N}_\varepsilon(p) \wedge dist(p,q) \leq nn\text{-}dist_{MinPts}(q)$

$\overset{\text{Def 2.10}}{\Longleftrightarrow} q \in \mathcal{N}_\varepsilon(p) \wedge dist(p,q) \leq \text{CORE}_{den}(q)$

$\Leftrightarrow q \in X$

□

Lemma 4.1 states that we can filter out a lot of points not belonging to $\text{CHANGE}_{den}(p)$ by computing only one range query around the update

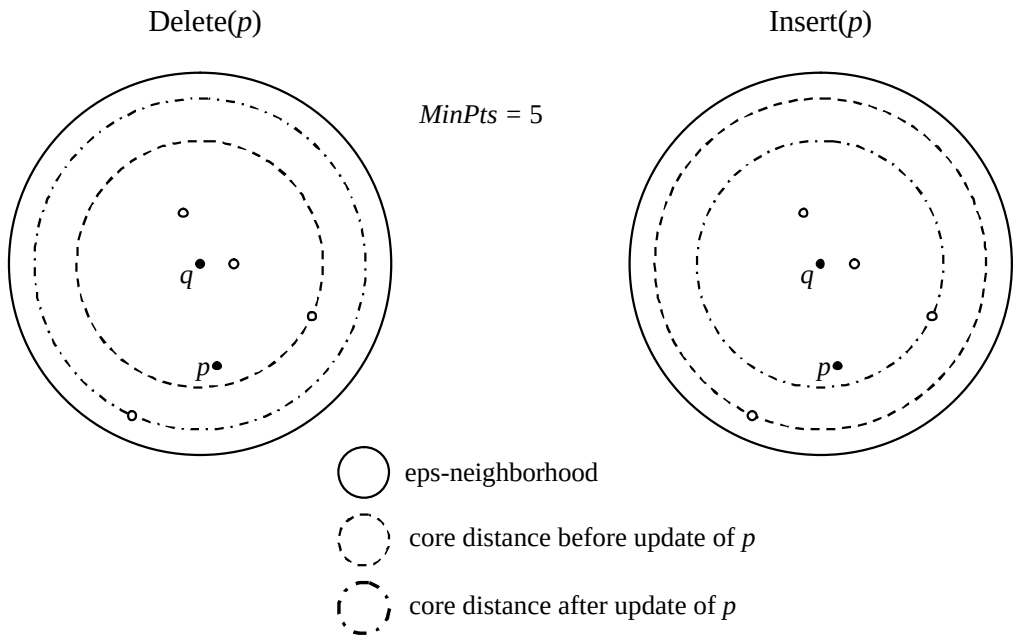


Figure 4.2: The core distance of q changes due to insertion/deletion of p.

point. In addition, we only have to test the points $q \in \mathcal{N}_\varepsilon(p)$ whether $q \in \text{REV}_{MinPts}(p)$. The idea is that for all $q \in \text{REV}_{MinPts}(p)$, it holds that $dist(p,q) \leq \text{CORE}_{den}(q)$. The change of the core distance of a point q due to an insertion/deletion of point p is illustrated in Figure 4.2. If p is inserted, the core distance of q decreases, whereas if p is deleted, the core distance of q increases.

The second step to determine the points in a cluster ordering affected by an update operation is to determine that points, the reachability distances of which are changing due to mutating core distances. A changing reachability distance may cause the violation of condition (2) in Definition 2.12. If a reachability distance of a point p decreases due to a changed core distance, p may move forward in the cluster ordering, otherwise, if the core distance of p increases, p may move backwards.

In the following, we say that q *comes before* p in the cluster ordering if $q.P < p.P$.

Definition 4.3 (predecessor in the cluster ordering)
Let CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$. The predecessor *of a point $p \in \mathcal{D}$* in the cluster ordering *CO, denoted by $\text{PRE}_{den}(p)$, is defined as follows:*

$$\text{PRE}_{den}(p) = \begin{cases} q & \text{if} \quad p.R = \text{REACHDIST}_{den}(q, p) \\ \text{UNDEFINED} & \text{if} \quad p.R = \infty \end{cases}$$

Intuitively, the predecessor of a point p is that point q in the cluster ordering from which p has been "reached" during the OPTICS run, i.e. p has been chosen at position $p.P$ because p had the minimum reachability distance of the not yet processed points to the already processed points, and this minimum reachability distance was determined by q. Obviously, that implies that $p.P > q.P$. If p has not been reached from any other point, its reachability distance $p.R$ in the cluster ordering is ∞, and thus, its predecessor is undefined.

Definition 4.4 (successors in the cluster ordering)
Let CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$. The successors *of a point $p \in \mathcal{D}$* in the cluster ordering *CO, denoted by $\text{SUC}_{den}(p)$, is defined as follows:*

$$\text{SUC}_{den}(p) = \{q \in CO \mid \text{PRE}_{den}(q) = p\}.$$

The successors of a point p include all points in the cluster ordering that have been "reached" from p, i.e. have been chosen at the according position in the cluster ordering because of their reachability distances w.r.t. p. Let us note that for each $q \in \text{SUC}_{den}(p)$, q in general comes after p in the cluster ordering, i.e. $p.P < q.P$. Since there are points in the cluster ordering that may not have a predecessor, there may also be points that do not have any successors.

In the following, CO_{old} denotes the original cluster ordering before the update. IncOPTICS aims at efficiently computing CO_{new}, the new (valid)

cluster ordering after insertion/deletion of a point u. We will create CO_{new} by performing a single pass over CO_{old}. During the creation, each point p keeps its three additional attributes $p.P$ (the position in the old/new ordering), $p.C$ (its new core level), and $p.R$ (the minimal reachability distance to all points already in CO_{new} which can be the original reachability distance assigned during the generation of CO_{old}, or a new value). In addition, we store the current predecessor $\text{PRE}_{den}(p)$ and the current successors $\text{SUC}_{den}(p)$.

4.2.2 Incremental Insertion of a Point

When inserting a point into a cluster ordering, the core distances of some points may decrease. As a consequence, some reachability distances may be not valid anymore and thus, some connections may be affected. As a consequence, further points (the successors of affected points) may be affected, too. The following definition of *potential successors* captures the points that may also be affected if a point p is affected due to an insertion.

Definition 4.5 (potential successors in the cluster ordering)
Let CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$. The potential successors *of a point $p \in \mathcal{D}$* in the cluster ordering CO, *denoted by $\text{SUC}^{pot}_{den}(p)$, includes all $q \in \mathcal{N}_\varepsilon(p)$, such that at least one of the following conditions hold:*

(1) *q has no predecessor, i.e. $\text{PRE}_{den}(q) = \text{UNDEFINED}$ or*

(2) *the original reachability distance of q is not smaller than the core distance of p, i.e. $q.R \geq p.C$ or*

(3) *the predecessor of q is not yet added to the new cluster ordering CO_{new}, i.e. $\text{PRE}_{den}(q) \notin CO_{new}$.*

The potential successors are formally defined as follows:

$$\text{SUC}^{pot}_{den}(p) = \{q \in CO \mid q \in \mathcal{N}_\varepsilon(p) \wedge (\text{PRE}_{den}(q) = \text{UNDEFINED} \vee q.R \geq p.C \vee \text{PRE}_{den}(q) \notin CO_{new}\}.$$

Lemma 4.2

Let CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$, and $p \in CO_{old}$. If p is affected during the reorganization, i.e. is part of the reorganization, and is inserted into CO_{new}, only the points in $\text{SUC}_{den}^{pot}(p)$ also need to be considered for a reorganization.

Proof.

(1) Let $x \in \text{SUC}_{den}^{pot}(p)$:

It follows from $x \in \mathcal{N}_{\varepsilon}(p)$ that x is reachable from p w.r.t. ε (and MinPts).

(1.1) $\text{PRE}_{den}(x) = \text{UNDEFINED}$:

Since x is reachable from p w.r.t. ε it is inserted into the seed list with $\text{PRE}_{den}(x) = p$. Thus, there may be a new connection between x and p that needs reorganization.

(1.2) $x.R \geq p.C$:

Since x is reachable from p w.r.t. ε and the current reachability of x is greater than the core-distance of p, it is possible, that x is also reachable w.r.t. an ε' which is smaller than the current reachability distance of x, i.e. $\varepsilon' < x.R$. If so, x is updated in the seed list including $\text{PRE}_{den}(x) = p$, and thus, there may be a connection between x and p that needs a reorganization.

(1.3) $\text{PRE}_{den}(x) \notin CO_{new}$:

Since x is reachable from p w.r.t. ε and the current predecessor of x is not yet in CO_{new}, it is possible, that x is the next point which must be inserted into CO_{new} with predecessor p, i.e. the predecessor of x may change.

(2) Let $x \notin \text{SUC}_{den}^{pot}(p)$:

(2.1) $x \notin \mathcal{N}_{\varepsilon}(p)$:

It is clear that x is not reachable from p w.r.t. ε, thus there cannot be a connection between x and p that needs reorganization.

(2.2) $\text{PRE}_{den}(x) \neq \text{UNDEFINED}$ and $x.R < p.C$ and $\text{PRE}_{den}(x).P \leq p.P$:

Since x has a predecessor that comes before p in CO_{new}, it is already in the seed list when p is processed. Obviously, x was inserted into the seed list at last after the processing of $\text{PRE}_{den}(x)$. From $x.R < p.C$ follows that x is already reachable w.r.t. a lower ε' than $\text{REACHDIST}_{den}(p, x)$ because $\text{REACHDIST}_{den}(p, x) \geq p.C$ and $x.R < p.C$. As a consequence, x will not be updated in the seed list when p is processed and thus, p cannot become x's

predecessor, i.e. there cannot be any connection between x and p that needs reorganization. □

To determine the potential successors of p we have to perform a range query around p. For objects that are not affected, i.e. which are taken over from CO_{old} without changes, we do not need to compute a range query. Let us note, that "without changes" do not imply that these points have the same position in CO_{old} and CO_{new}. Lemmata 4.1 and 4.2 are important to determine the points that may be affected by an insertion. Only the points in $\textsc{Change}_{den}(u)$ and recursively the potential successors of affected points may be affected and need a range query for reorganizing their reachability connectivities.

Algorithm Insert

Now we are able to develop an incremental algorithm for the insertion of a point u. The pseudo code of the incremental insert algorithm is depicted in Figure 4.3. As mentioned above, we assume that for each $o \in CO_{old}$ the predecessor $\textsc{Pre}_{den}(o)$ and the set of successors $\textsc{Suc}_{den}(o)$ have been correctly determined. $\textsc{Pre}_{den}(o)$ can be computed on the fly during the OPTICS run by adopting the method OrderedSeeds::update (cf. Figure 2.10) as depicted in Figure 4.4. The idea of the adoption is to set the predecessor of an inserted object q to the object o from which q is actually reached. If q is decreased in the seed list, its predecessor must be updated accordingly. $\textsc{Suc}_{den}(o)$ can be computed after the OPTICS run from the predecessor information of the points in the cluster ordering.

In the first step of the insertion of u, the core distances of each $o \in \textsc{Change}_{den}(p)$ are updated and u is inserted into the seed list OrderedSeeds with a reachability distance $p.R = \infty$. This is because it is not yet clear, from which object u is reached in CO_{new}.

After that, the reorganization is performed, imitating the original OPTICS algorithm. We manage the points that need reorganization in the seed list. In each iteration of the reconstruction loop, we compare the next not yet

```
algorithm insert(Object u, ClusterOrdering CO_old)
  // all points in CO_old are marked as not yet handled
  u.P := n + 1;
  u.C := CoreDist_den(u);
  CO_new := empty cluster ordering;
  for each o ∈ Change_den(u) do
    update the core distance of o;
  end for
  insert u into OrderedSeeds with reachability distance ∞;
  while CO_old contains unhandled points or OrderedSeeds ≠ ∅ do
    c := first not yet handled object in CO_old;
    s := first not yet handled object in OrderedSeeds;
    if s.R > c.R or (s.R = c.R and s.P > c.P) then
      append c to CO_new;
    else
      remove s from OrderedSeeds;
      append s to CO_new;
    end if
    l := the object recently appended to CO_new;
    mark l as handled;
    if l has been chosen from OrderedSeeds or l ∈ Change_den(u) then
      OrderedSeeds.update(Suc^pot_den(l), l);
    else
      if u is not yet handled and l.C ≤ ε and dist(u,l) ≤ ε then
        OrderedSeeds.update({u}, l);
      end if
    end if
  end while
```

Figure 4.3: Algorithm insert for IncOPTICS.

handled point c in CO_{old} with the first point s in OrderedSeeds. The point of c and s which has the smaller reachability distance value is appended to CO_{new}. If the reachability distance values of both points are equal, i.e. $c.R = s.R$, we append that point having the smaller position value $\_.P$ to CO_{new}.

After the insertion of a point l in the new cluster ordering CO_{new}, we have to update the seed list OrderedSeeds. This is done by the method OrderedSeeds::update depicted in Figure 4.4. As mentioned above, this method is an adoption of the original method presented in Section 2.3. If the recently processed point l is derived from the original cluster ordering CO_{old}, we have to test whether the update point u has been already processed. If

```
method OrderedSeeds::update(SetOfObjects objects, Object o)
  for each q ∈ objects do
    if q is not yet processed then
      if o = null then
        rdist := ∞;
      else
        rdist := max{o.C, dist(o, q)};
      end if
      if q is already contained in OrderedSeeds then
        if q.R > rdist then
          q.R := rdist;
          decrease(q);
          PRE_den(q) := o
        end if
      else
        q.R := rdist;
        insert(q);
        if rdist < ∞ then
          PRE_den(q) := o;
        end if
      end if
    end if
  end for
```

Figure 4.4: IncOPTICS: adopted method OrderedSeeds::update.

u has not been processed and $l.C \neq \infty$ and $dist(l,u) \leq \varepsilon$, u would have been inserted/updated in OrderedSeeds in the original OPTICS run. Thus, l may be a potential predecessor of u and OrderedSeeds has to be updated accordingly. Other connections are not affected, since we store the points that need reorganization in the seed list. If l is derived from OrderedSeeds or from $\textsc{Change}_{den}(u)$, some connections may need reorganization. Thus, all not yet processed potential successors $x \in \textsc{Suc}^{pot}_{den}(l)$ have to be inserted/updated in the seed list.

The reorganization stops if the original cluster ordering CO_{old} does not contain unprocessed points any more and the seed list is empty.

Correctness

Lemma 4.3

The incremental insert algorithm is correct, i.e. produces a valid cluster

ordering w.r.t. Definition 2.12.

Proof. *We have to show that CO_{new} is valid w.r.t. Definition 2.12 after the insertion of u. Obviously, condition (1) holds for CO_{new}: the correct core distance is assigned to each point. Conditions (2) and (3) in Definition 2.12, however, need some further verification.*

In order to get a valid cluster ordering, we have to ensure that at each step of the reorganization, we choose that point having the minimum reachability distance to the already processed points. Obviously, only points that are contained in the ε-neighborhood of at least one point coming before position i have to be considered for the next free position i in CO_{new}.

At the beginning of the generation of CO_{new}, only u is in OrderedSeeds *with $u.R = \infty$. Thus, the first object in CO_{new} will be the first object o in CO_{old} because $o.R = \infty = u.R$ but $o.P = 1 < n + 1 = u.P$.*

During the reorganization, we choose that not yet processed point either from CO_{old} or OrderedSeeds *having the minimum reachability distance. If both points have equal reachability distances, we choose that point coming first in CO_{old} (maintaining the non-deterministic order generated by the original OPTICS run). Thus, the critical aspect for the correctness is the correct maintenance of the current reachability distances of the points w.r.t. that points already added to CO_{new}. As we have seen from Lemma 4.2, only the reachability distances of the potential successors of affected points may change because some connectivities are reorganized. This maintenance is worked out in the last part of the algorithm, where it is decided how to update the seed list, depending on the last point l added into CO_{new}.*

If l has been chosen from CO_{old} and $l \notin \textsc{Change}_{den}(u)$, we need no reorganization, i.e. the connectivities within the old cluster ordering are locally conserved. The only problem may be that l is a potential predecessor of u. This can only be true if u is not yet processed and the core distance of l is defined (i.e. $l.C \leq \varepsilon$) and $u \in \mathcal{N}_\varepsilon(l)$ (i.e. $dist(l, u) \leq \varepsilon$). If so, we have to update the predecessor of u in OrderedSeeds*.*

If l has been chosen from OrderedSeeds *or $l \in \textsc{Change}_{den}(u)$, a local*

reorganization takes place because in the first case, the old cluster ordering may be altered locally, and in the second case, successors of l may need reorganization due to the changed core distance of l. In both cases, all not yet processed potential successors $\text{SUC}^{pot}_{den}(l)$ *have to be inserted into* OrderedSeeds *with current predecessor l or updated (if they are already in* OrderedSeeds*) in terms of their current predecessors. Due to Lemma 4.2, no other point have to be considered, i.e. inserted into the seed list.* □

4.2.3 Incremental Deletion of a Point

When deleting a point from a cluster ordering, the core distances of some points may increase. Again, this may affect the reachability distances of some points, i.e. some connections need reorganization. The reorganization of some connections may result in further affected points. The following definition of *recursive successors* captures the points that may need reorganization if a point p has been reorganized due to a deletion.

Definition 4.6 (recursive successors in the cluster ordering)
Let CO be a cluster ordering of $\mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$. The recursive successor *of a point $p \in \mathcal{D}$* in the cluster ordering *CO, denoted by* $\text{SUC}^{rec}_{den}(p)$*, is defined recursively:*

(1) $o \in \text{SUC}_{den}(p) \Rightarrow o \in \text{SUC}^{rec}_{den}(p)$

(2) $q \in \text{SUC}^{rec}_{den}(p) \wedge o \in \text{SUC}_{den}(q) \Rightarrow o \in \text{SUC}^{rec}_{den}(p)$

Algorithm Delete

Now we are able to develop an incremental algorithm for the deletion of a point u. The pseudo code of the incremental delete algorithm is depicted in Figure 4.5. We again assume that for each $o \in CO_{old}$ the predecessor $\text{PRE}_{den}(o)$ and the set of successors $\text{SUC}_{den}(o)$ have been correctly determined.

```
algorithm delete(Object u, ClusterOrdering CO_old)
  // all points in CO_old are marked as not yet handled
  mark u as handled;
  CO_new := empty cluster ordering;
  rs := ∅;
  for each o ∈ CHANGE_den(u) do
    update the core distance of o;
    insert SUC^rec_den(o) into rs;
  end for
  OrderedSeeds.update(rs, null);
  while CO_old contains unhandled points or OrderedSeeds ≠ ∅ do
    c := first not yet handled object in CO_old not contained in rs;
    s := first not yet handled object in OrderedSeeds;
    if s.R ≤ c.R or PRE_den(c) is not yet handled then
      remove s from OrderedSeeds;
      append s to CO_new;
    else
      append c to CO_new;
    end if
    l := the object recently appended to CO_new;
    mark l as handled;
    if l.C ≤ ε then
      OrderedSeeds.updateAll(l, ε);
      OrderedSeeds.update(SUC_den(l), l);
    end if
  end while
```

Figure 4.5: Algorithm delete for IncOPTICS.

In the first step of the delete method, u is marked as handled. This ensures that u is not inserted into CO_{new}. In addition, for each $o \in \text{CHANGE}_{den}(p)$ the core distance of each o is updated and its recursive successors $\text{SUC}^{rec}_{den}(o)$ are inserted into the seed list OrderedSeeds. For the insertion, we use the method OrderedSeeds::update from Figure 4.4 and the **null**-value for the second parameter, ensuring that the reachability distance of each inserted point is set to ∞ and the predecessor is set to **null** (i.e. UNDEFINED). Thus, we have placed all recursive successors of $o \in \text{CHANGE}_{den}(u)$ into the seed list with a reachability distance of ∞ because we do not yet know from which points they will be reached in CO_{new}.

After that, the reorganization is worked out, again imitating the original OPTICS algorithm and managing the points that need reorganization in the seed list. In each iteration of the reconstruction loop, we compare the reach-

```
method OrderedSeeds::updateAll(Object o, Real ε)
  for each q ∈ OrderedSeeds do
    if dist(o,q) ≤ ε then
      rdist := max{o.C, dist(o,q)};
      if rdist < q.R then
        q.R := rdist;
        decrease q;
        PRE_den(q) := o;
      end if
    end if
  end for
```

Figure 4.6: IncOPTICS: method OrderedSeeds::updateAll.

ability distance of the next not yet handled point c in CO_{old} which is not contained in $\text{CHANGE}_{den}(u)$ with that of the first point s in OrderedSeeds. If the predecessor of c ($\text{PRE}_{den}(c)$) is not yet processed (i.e. inserted into CO_{new}), c cannot be taken from CO_{old} and appended to CO_{new}. Otherwise, the point of c and s, having the smaller reachability distance value, is appended to CO_{new}.

After the insertion of a point l in the new cluster ordering CO_{new}, we have to update OrderedSeeds. This is done by the method OrderedSeeds::update depicted in Figure 4.4 and OrderedSeeds::updateAll depicted in Figure 4.6. If the recently processed point l is a core object w.r.t. the generic ε-value, then l can become a predecessor of all points that are still in the seed list. Thus, we have to update all points in OrderedSeeds which is worked out by the method OrderedSeeds::updateAll. In addition, all successors of l ($\text{SUC}_{den}(l)$) have to be inserted into OrderedSeeds because they may be affected by changing reachability distances or affected predecessors.

The reorganization stops if the original cluster ordering CO_{old} does not contain unprocessed points any more and the seed list is empty.

Correctness

Lemma 4.4

The incremental delete algorithm is correct, i.e. produces a valid cluster

ordering w.r.t. Definition 2.12.

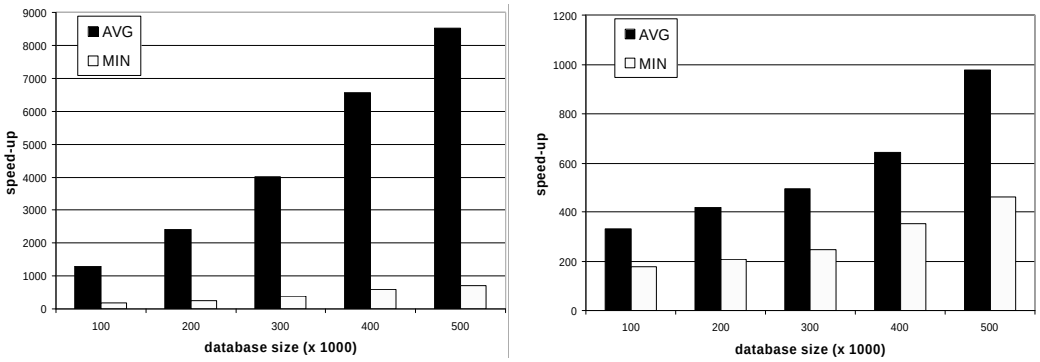
Proof. *analogously to Lemma 4.3* □

4.2.4 Extensions for Bulk Updates

Both the insert as well as the delete methods perform one pass over the original cluster ordering, i.e. have a runtime complexity of at least $O(n)$. However, the great benefit is that both incremental updates save as many range queries as possible. In addition, the incremental algorithms for insertion and deletion of a single point can be easily extended to work on a bulk of insert/delete points. The difference to the methods insert and delete is that the core distances of all points $o \in \text{CHANGE}_{den}(u)$ (for all update points u) have to be updated first. After that — in case of insertion — we simply insert all update points into OrderedSeeds instead of only one update point. The rest of the bulk insertion is analogously to the single insertion. In case of a deletion, we have to insert the recursive successors $\text{SUC}^{rec}_{den}(o)$ for all $o \in \text{CHANGE}_{den}(u)$ (for all update points u) into OrderedSeeds, analogously. The reorganization loop is also identical to the single update procedure. Thus, to insert/delete a set of n_u points should be rather efficient because we do not have to perform n_u passes over the original cluster ordering but can work out the reorganization for the n_u insertions/deletions in one single pass. This is an important advantage because bulk updates are a realistic scenario in batch mode systems such as data warehouses, i.e. databases, in which the updates are not performed immediately but are collected and applied in a batch mode (e.g. over night). Obviously, the bulk update algorithms are also correct.

4.3 Experimental Evaluation

We evaluated the efficiency of IncOPTICS in comparison to the original OPTICS algorithm on several synthetic as well as on a real-world data set. The synthetic data sets contain a diverse number of 2-dimensional feature



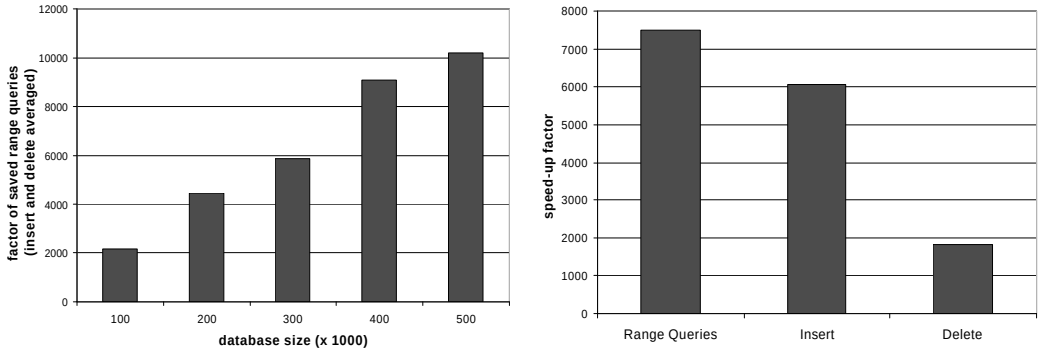
(a) Insert **(b)** Delete

Figure 4.7: Runtime speed-up factors of IncOPTICS vs. OPTICS.

vectors from $n = 100,000$ to $n = 500,000$. The real-world data set contains around 100,000 feature vectors representing TV snapshots encoded by 64-dimensional color histograms. We used an X-Tree [BKK96] for speeding up the range queries for both IncOPTICS and the original OPTICS algorithm. Since both algorithms benefit from the index structure, we did not consider the index creation time in our runtime experiments. For each data set, we performed 500 random inserts and 500 random deletions.

IncOPTICS gained impressing speed-up factors compared to the non-incremental version of OPTICS. Figure 4.7 depicts the average and minimum runtime speed-up factors in case of a single insertion/deletion w.r.t. the size of the database. It can be observed that with growing database size, the runtime gain is increasing from an average speed-up factor of 1.300 ($n = 100,000$) to factor 8.500 ($n = 500,000$) in case of insertion. In case of deletion, the speed-up factors are slightly lower — 330 ($n = 100,000$) to nearly 1,000 ($n = 500,000$). However, even in the worst case, IncOPTICS achieved at least speed-up factors between 170 and 700. The reason for the less high speed-up factors in case of deletions is that for more objects a new predecessor need to be determined. Let us note that in case of a bulk update, this effect is less significant (see below).

The main reason for this large speed-up is depicted in Figure 4.8(a) showing the average number of range queries saved by IncOPTICS during one sin-



(a) Synthetic data **(b)** TV data

Figure 4.8: Results of IncOPTICS on synthetic and real-world TV data.

gle update operation. IncOPTICS achieves its significant performance gain over OPTICS by saving from 2,000 to more than 10,000 times of the range queries which are necessary for the original OPTICS run. This empirically shows that the strategy of limiting the reorganization to a predefined part of the cluster ordering is usually much more efficient than recomputing the cluster ordering from scratch.

The experimental results on the TV data set, depicted in Figure 4.8(b), confirms that observation. The left bar in Figure 4.8(b) illustrates the factor of range queries saved by IncOPTICS compared to OPTICS on the TV data (factor 7,500). The two other bars show the average speed-up factors for 500 random insertion (factor 6,000) and 500 random deletion (factor 1,800) that IncOPTICS yields over OPTICS.

Let us note that the use of an index such as the X-Tree favors the original OPTICS because it accelerates the computation of the range queries. If the range queries are computed on top of the sequential scan, IncOPTICS may most likely yield even higher speed-up factors since it saves a significant amount of queries. This is especially important for high dimensional data sets where the performance boost of most index structures usually deteriorate.

To test the performance of the bulk mode of IncOPTICS, we used the synthetic data set containing 200,000 points in a 2-dimensional feature space. We performed bulk update operations using sets of 500 to 4,000 update

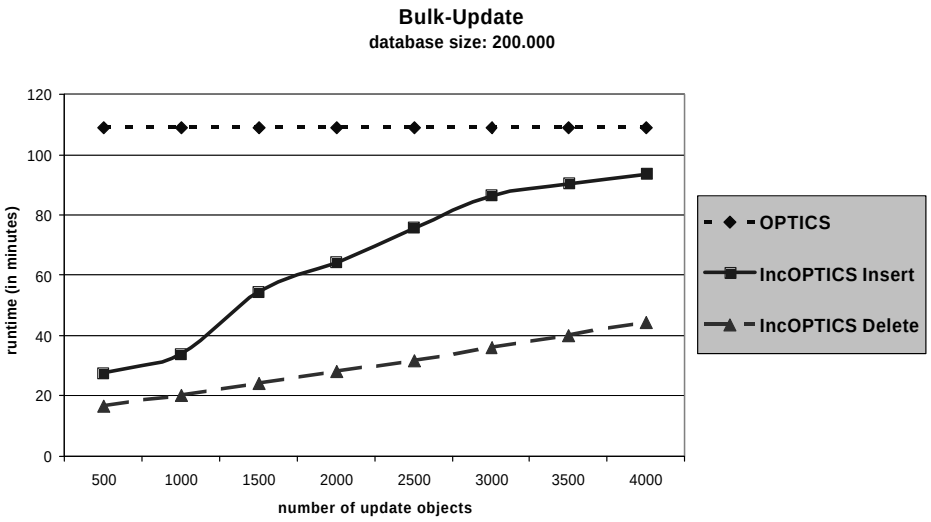


Figure 4.9: Comparison of bulk IncOPTICS vs. OPTICS.

objects. A runtime comparison of the bulk update runs are depicted in Figure 4.9. It can be seen that even when inserting or deleting 20% of the database, the runtime of IncOPTICS is still significantly smaller than the original OPTICS algorithm. In addition, it can be observed that deletions can be worked out much faster in the bulk mode than in the single update mode compared to insertions. The reason for this behavior is indicated in the following. When inserting a point p of the update set, we have to check each remaining point q in the update set if $p \in \textsc{Change}_{den}(q)$. If this is the case, q has to be inserted into the seed list. Obviously, the higher the number of update points, the higher the probability that $p \in \textsc{Change}_{den}(q)$.

4.4 Summary

Incrementally maintaining the cluster hierarchy computed by OPTICS is a mandatory requirement for the BOSS browsing tool, desired to work in dynamic database scenarios. In this chapter, we proposed an incremental variant of the OPTICS algorithm called IncOPTICS that efficiently handles insertions and deletions of points.

The performance of IncOPTICS is evaluated on synthetic and real-world data sets. On the average, IncOPTICS yields rather significant speed-up factors (e.g. 8,500 for an insertion and 1,000 for a deletion, both on a database of 500,000 points) over OPTICS. Due to our experimental results, this performance gain is achieved — although IncOPTICS performs a single pass over the cluster ordering for reconstruction — by saving unnecessary range queries during the reorganization of the cluster ordering.

A further advantage of the proposed incremental OPTICS variant is that, by applying very simple extensions, it can handle bulk updates rather efficiently. In the presented experiments, significant speed-up factors can still be achieved when inserting/deleting 20% of a database of 200,000 points. It can be expected that for larger databases the results further improve. Thus, the incorporation of IncOPTICS provides BOSS with the applicability to large dynamic databases.

Chapter 5

Cluster Recognition and Representation

The BOSS prototype is supposed to enable smart and comfortable browsing through a hierarchy of clusters. Thus, a second major challenge for density-based hierarchical clustering, identified in Section 3.2, is solid cluster recognition and intuitive cluster representation. In this chapter, we first address the task of cluster recognition, i.e. the extraction of meaningful clusters from a density-based cluster ordering, in Section 5.1. After a discussion of related work about cluster extraction from hierarchical representations, we introduce a novel approach, called GradientClustering to extract hierarchies of clusters from a cluster ordering. Then, we present novel approaches for the problem of finding meaningful cluster representatives in Section 5.2. A short summary in Section 5.3 concludes this chapter. The basic ideas contained in this chapter have been published in [BKK+03], [BJK+03], and [BKKP04].

5.1 Cluster Recognition

5.1.1 Related Work

To the best of our knowledge, there are only two methods for automatic cluster extraction from hierarchical representations such as reachability plots or dendrograms, both are also based on reachability plots. Since clusters are represented as valleys (or dents) in the reachability plot, the task of automatic cluster extraction is to identify significant valleys.

The first approach proposed in [ABKS99], called ξ-clustering, is based on the steepness of the borders of valleys in the reachability plot. Each cluster starts with a so-called steep downward area. A steep downward area is an interval in the cluster ordering where the reachability distances of points strictly decreases and which is flanked by points whose reachability distance is ξ% higher than that of their successors. Each cluster ends with a so-called steep upward area which is defined analogously to steep downward areas. The parameter ξ, specifying the steepness of these areas, is the input parameter of this cluster recognition method. The method suffers from the fact that this input parameter is difficult to understand and hard to determine. Rather small variations of the value ξ often lead to drastic changes of the resulting clustering hierarchy. As a consequence, this method is unsuitable for our purpose of automatic cluster extraction.

The second approach was proposed recently by Sander et al. [SQL+03]. The authors describe an algorithm called `cluster_tree` that automatically extracts a hierarchical clustering from a reachability plot and computes a cluster tree. It is based on the idea that *significant* local maxima in the reachability plot separate clusters. Two parameters are introduced to decide whether a local maximum is significant: The first parameter specifies the minimum cluster size, i.e. how many objects must be located between two significant local maxima. The second parameter specifies the ratio between the reachability distance of a significant local maximum m and the average reachability distances of the regions to the left and to the right of m. The authors in [SQL+03] propose to set the minimum cluster size to 0.5% of the

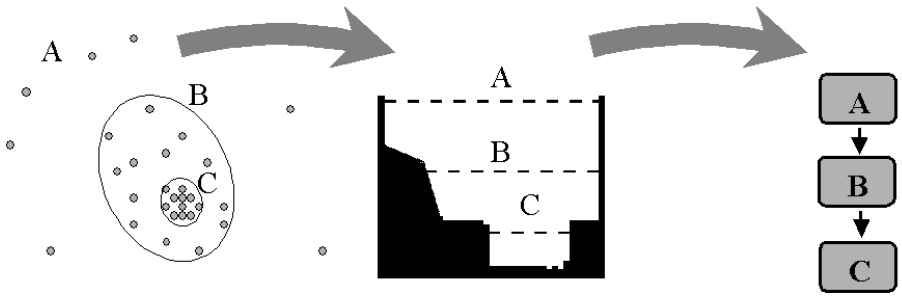


Figure 5.1: Sample nested clusters: data space (left); reachability plot (middle); cluster hierarchy (right)

data set size and the second parameter to 0.75. They empirically show that this default setting approximately represents the requirements of a typical user.

Although the `cluster_tree` method is very intuitive and rather suitable for automatic cluster extraction from reachability plots, it has one major drawback. Many real-world data sets consist of nested clusters, i.e. clusters each consisting of exactly one smaller sub-cluster (cf. Figure 5.1).

Since the algorithm `cluster_tree` runs through a list of all local maxima (sorted in descending order of reachability distance) and decides at each local maximum m whether m is significant to split the objects to the left of m and to the right of m into two clusters, the algorithm cannot detect such nested clusters. These clusters cannot be split by a significant maximum. Figure 5.1 illustrates this fact. The nested cluster A contains the sub-cluster B which itself contains the sub-cluster C (the clusters are indicated by dashed lines in the reachability plot). The algorithm `cluster_tree` will only find cluster A since there are no local maxima to split clusters B and C. This makes the algorithm unsuitable for the intended BOSS system. Obviously, the `cluster_tree` algorithm was designed to help an unexperienced user to get a quick overview of the most significant parts of the cluster hierarchy and to prevent the user from getting overwhelmed by potentially uninteresting details. However, in the desired BOSS system — especially for the similarity

search application — we want to extract the details of the cluster hierarchy and not missing out some small clusters or the split of a larger cluster into smaller sub-clusters.

In addition, also the ξ-clustering will only detect one of the clusters A, B or C depending on the ξ-parameter but will fail to detect the cluster hierarchy.

Thus, for the purposes of BOSS, we need a new cluster recognition algorithm that should meet the following requirements:

- It should detect all kinds of sub-clusters, including nested sub-clusters.
- It should create a clustering structure which is close to the one an experienced user would manually extract from a given reachability plot.
- It should allow an easy integration into the OPTICS algorithm. We do not want to apply an additional cluster recognition step after the OPTICS run is completed. In contrast, the hierarchical clustering structure should be created on-the-fly during the OPTICS run without causing any considerable additional cost.
- It should be integrable into the incremental version of OPTICS.

5.1.2 Gradient Clustering

In this section, we introduce our new GradientClustering algorithm which fulfills all of the above mentioned requirements. The idea behind our new cluster extraction algorithm is based on the concept of *inflection points*. During the OPTICS run we decide for each point added to the result set, i.e. the reachability plot, whether it is an inflection point or not. If it is an inflection point, we might be at the start or at the end of a new cluster. We store the possible starting points of the sub-clusters in a stack, called $startPts$. This stack consists of pairs $(o.P, o.R)$. Our GradientClustering algorithm can easily be integrated into OPTICS and is described in full detail after we have formally introduced the new concept of *inflection points*.

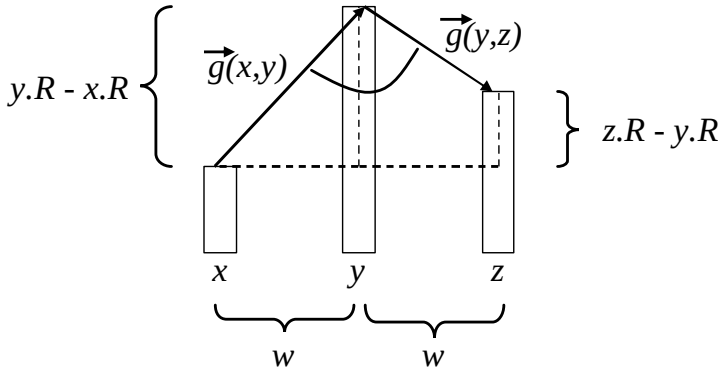


Figure 5.2: Gradient vectors $\vec{g}(x, y)$ and $\vec{g}(y, z)$ of objects x, y and z adjacent in the cluster ordering.

In the following, we assume that CO is a cluster ordering as defined in Definition 2.12. We call two objects $o_1, o_2 \in CO$ *adjacent in* CO if $o_2.P = o_1.P + 1$ or *vice versa*. Let us recall that $o.R$ is the reachability distance of $o \in CO$ assigned by OPTICS while generating CO. For any two objects $o_1, o_2 \in CO$ adjacent in the cluster ordering, we can determine the gradient of the reachability distances $o_1.R$ and $o_2.R$. The gradient can easily be modeled as a 2D vector where the y-axis measures the reachability distances ($o_1.R$ and $o_2.R$) in the ordering, and the x-axis represents the ordering of the objects. If we assume that each object in the ordering is separated by width w, the gradient of o_1 and o_2 is the vector

$$\vec{g}(o_1, o_2) = \begin{pmatrix} w \\ o_2.R - o_1.R \end{pmatrix}.$$

An example for a gradient vector of two objects x and y adjacent in a cluster ordering is depicted in Figure 5.2.

Intuitively, an inflection point should be an object in the cluster ordering where the gradient of the reachability distances changes significantly. This significant change indicates a starting or an end point of a cluster.

Let $x, y, z \in CO$ be adjacent, i.e. $x.P + 1 = y.P = z.P - 1$. We can now measure the differences between the gradient vector $\vec{g}(x, y)$ and $\vec{g}(y, z)$ by computing the cosinus function of the angle between the vectors $\vec{g}(x, y)$ and $\vec{g}(z, y) = (-w, y.R - z.R)^{\mathbf{T}}$. The cosinus of this angle is equal to -1

if the angle is 180°, i.e. the vectors have the same direction. On the other hand, if the gradient vectors differ a lot, the angle between them will be clearly smaller than 180° and thus the cosinus will be significantly greater than -1. This observation motivates the concepts of inflection index and inflection points:

Definition 5.1 (inflection index) *Let CO be a cluster ordering and $x, y, z \in CO$ be objects adjacent in CO. The* inflection index *of y, denoted by $II(y)$, is defined as the cosinus of the angle between the gradient vector of x, y ($\vec{g}(x, y)$) and the "inverse" gradient vector of y, z ($\vec{g}(z, y)$), formally:*

$$II(y) = \cos \varphi_{(\vec{g}(x,y),\vec{g}(z,y))} = \frac{-w^2 + (y.R - x.R)(y.R - z.R)}{\|\vec{g}(x, y)\| \, \|\vec{g}(z, y)\|},$$

where $\|\vec{v}\| := \sqrt{v_1^2 + v_2^2}$ is the length of the vector $\vec{v}$.

Definition 5.2 (inflection point) *Let CO be a cluster ordering and $x, y, z \in CO$ be objects adjacent in CO and let $t \in \mathbb{R}$ ($t \in [-1, 1]$). Object y is an* inflection point *iff*

$$II(y) > t.$$

The concept of inflection points is suitable to detect objects in CO which are interesting for extracting clusters.

Definition 5.3 (gradient determinant) *Let CO be a cluster ordering and $x, y, z \in CO$ be objects adjacent in CO. The* gradient determinant *of the gradients $\vec{g}(x, y)$ and $\vec{g}(z, y)$ is defined as*

$$gd(\vec{g}(x, y), \vec{g}(z, y)) := \begin{vmatrix} w & -w \\ x.R - y.R & z.R - y.R \end{vmatrix}.$$

If x, y, z are clear from the context, we use the short form $gd(y)$ for the gradient determinant $gd(\vec{g}(x, y), \vec{g}(z, y))$.

The sign of $gd(y)$ indicates whether $y \in CO$ is a starting point or end point of a cluster. In fact, we can distinguish the following two cases which are visualized in Figure 5.3:

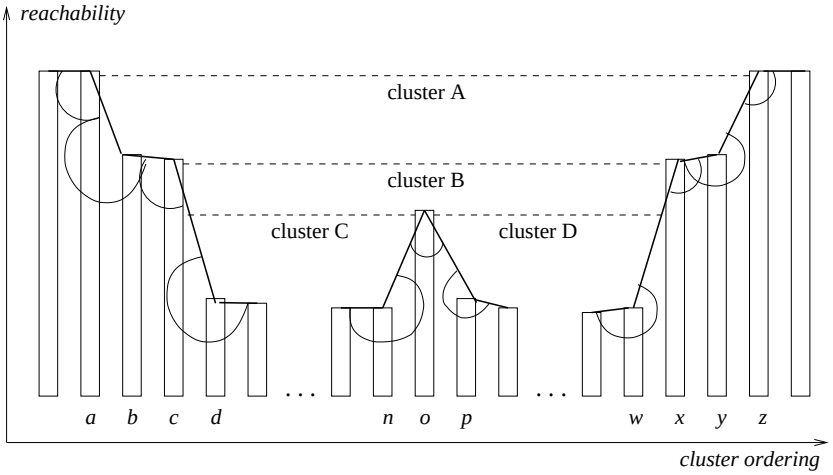


Figure 5.3: Illustration of inflection points measuring the angle between the gradient vectors of objects adjacent in the ordering.

- $II(y) > t$ and $gd(y) > 0$:
  Object y is either a starting point of a cluster (e.g. object a in Figure 5.3) or the first object outside of a cluster (e.g. object z in Figure 5.3).

- $II(y) > t$ and $gd(y) < 0$:
  Object y is either an end point of a cluster (e.g. object n in Figure 5.3) or the second object inside a cluster (e.g. object b in Figure 5.3).

Let us note that a local maximum $m \in CO$, which is the cluster separation point in [SQL+03], is a special form of the first case (i.e. $II(m) > t$ and $gd(m) > 0$).

The threshold t is independent from the absolute reachability distances of the objects in CO. The influence of t is also very comprehensible because if we know which values for the angles between gradients are interesting, we can easily compute t. For example, if we are interested in angles $< 120°$ and $> 240°$, we set $t = \cos 120° = -0.5$.

Obviously, the gradient clustering algorithm is able to extract narrowing clusters. Our experimental comparisons with the `cluster_tree` and ξ-clustering methods in Section 5.1.3 confirm this observation.

The pseudo code of the GradientClustering algorithm is depicted in Figure 5.11 on page 86. Initially, the first object of the cluster ordering CO is pushed to the stack of starting points $startPts$. Whenever a new starting point is found, it is pushed to the stack. If the current object is an end point, a new cluster is created, containing all objects between the starting point on top of the stack and the current end point. Starting points are removed from the stack if their reachability distance is lower than the reachability distance of the current object. Clusters are created as described above for all removed starting points as well as for the starting point which remains in the stack.

The input parameter $MinPts$ determines the minimum cluster size and the parameter w influences the gradient vectors and proportionally depends on the reachability distances of the objects in CO. In fact, a good solution is to normalize all reachability distance values within the interval $[0, 1]$ and set $w = 0.5$. The normalization can be achieved on the fly during the OPTICS run by computing $o.R_{norm} := o.R/\varepsilon$ for each $o \in CO$, where the reachability distance after a jump (i.e. for an object for which no predecessor exists) is set to ε instead of ∞. We used this proceeding throughout all our experiments.

Let us note that the GradientClustering algorithm can also easily integrated into IncOPTICS, the incremental version of OPTICS proposed in Chapter 4. The reason for this is that IncOPTICS constructs the new (updated) cluster ordering step by step similar to the original OPTICS algorithm.

5.1.3 Experimental Evaluation

We evaluated both the effectiveness and efficiency of our approach using two real-world test data sets. The first one contains approximately 200 CAD objects from a German car manufacturer, and the second one is a sample of the Protein Databank (PDB) [BWF+00] containing approximately 5,000 protein structures. We tested on a workstation featuring a 1.7 GHz CPU and 2 GB RAM.

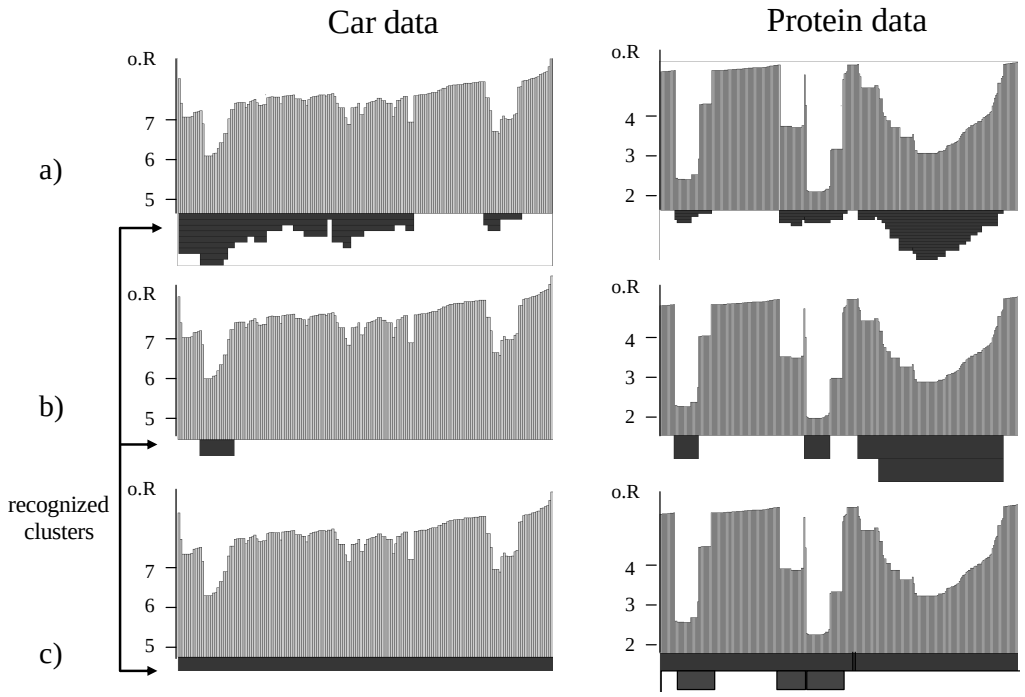


Figure 5.4: Clusters found on car parts and proteins by: a) GradientClustering, b) ξ-Clustering, c) `cluster_tree`

Effectivity

Both the Car and the Protein data set exhibit the commonly seen quality of unpronounced but nevertheless to the observer clearly visible clusters. The corresponding reachability plots of the two data sets are depicted in Figure 5.4.

Figure 5.4c shows that the `cluster_tree`-algorithm does not find any clusters at all in the Car data set with the suggested default ratio-parameter of 75% [SQL+03]. In order to detect clusters in the CAR data set, we had to adjust the ratio-input parameter to 95%. In this case, the `cluster_tree`-algorithm detected some clusters but missed out on some other important clusters and did not detect any cluster hierarchies at all. If we have rather high reachability distances, e.g. values between 5-7 as in Figure 5.4 for the Car data set, the ratio-parameter for the `cluster_tree`-algorithm should be higher than for smaller values. In the case of the Protein data set, we detected

	200 car parts	5,000 protein molecules
ξ-clustering	0.221 s	5.057 s
`cluster_tree`	0.060 s	1.932 s
Gradient Clustering	0.310 s	3.565 s

Table 5.1: CPU time for cluster recognition.

several clusters with the default parameter setting, but again missed out on some important clusters. Generally, in cases where a reachability graph consists of rather high reachability distances or does not present peaks at all, but clusters are formed by smooth troughs in the waveform, this cluster recognition algorithm is unsuitable. Furthermore, it is inherently unable to detect nested clusters where a cluster has one sub-cluster of increased density (cf. Figure 5.1).

On the other hand, the ξ-clustering approach successfully recognizes some clusters while also missing out on significant sub-clusters (cf. Figure 5.4b). This algorithm has some trouble recognizing cluster structures with a significant differential of "steepness". For instance, in Figure 5.1 it does not detect the nested cluster B inside of cluster A because it tries to create steep down-areas containing as many points as possible. Thus, it will merge the two steep edges if their steepness exceeds the threshold ξ. On the other hand, it is able to detect cluster C within A.

Finally, we look at our new *Gradient Clustering* algorithm. Figure 5.4a shows that the recognized cluster structure is close to the intuitive one, which an experienced user would manually derive. Clusters which are clearly distinguishable and contain more than $MinPts$ elements are detected by this algorithm. Not only does it detect a lot of clusters, but it also detects a lot of meaningful cluster hierarchies consisting of nested sub-clusters.

To sum up, in all our tests the GradientClustering algorithm detected much more clusters than the other two approaches without producing any redundant and unnecessary cluster information.

Efficiency

In all our tests we first created the reachability plots and then applied the algorithms for cluster recognition and representation. Let us note that we can also have integrated the GradientClustering into the OPTICS run without causing any noteworthy overhead.

The overall runtimes for the three different cluster recognition algorithms are depicted in Table 5.1. Our new GradientClustering algorithm does not only produce the most meaningful results, but also in sufficiently short time. This is due to its runtime complexity of $O(n)$.

5.2 Cluster Representation

Many partitioning clustering algorithms are known to use means or medoids as cluster representatives. The mean — also called centroid — is not suitable for cluster representation, since it is usually an artificial object not contained in the database. For complex objects (e.g. CAD parts, proteins), it may be quite difficult or even impossible to display a centroid. The medoid of a cluster C is the closest object to the mean of all objects in C. If $k > 1$ representatives should be generated, one could simply choose the k closest objects to the centroid of C as representatives.

The choice of medoids as cluster representative is somehow questionable. Obviously, if C is not of convex shape, the medoid is not really meaningful.

An extension of this approach coping with the problem of clusters with non-convex shape is the computation of k medoids by applying a k-medoid clustering algorithm to the objects in C. The clustering using a k-medoid algorithm is rather efficient due to the expectation that the clusters are much smaller than the whole data set. This approach can also be easily extended to cluster hierarchies. At any level the k-medoid clustering algorithm can be applied to the merged set of objects from the child clusters or — due to performance reasons — merge the medoids of child clusters and apply k-medoid clustering on this merged set of medoids. However, the questionable

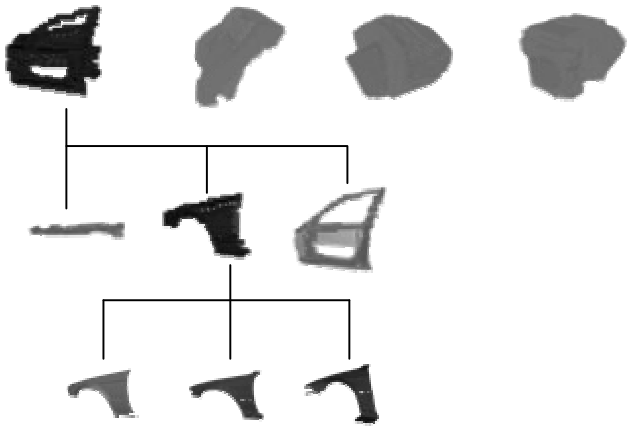


Figure 5.5: Representing clusters by superimposing all contained objects.

representative power of medoids for clusters of arbitrarily shaped clusters still remains.

In this section, we present two new approaches to determine representative objects for clusters computed by OPTICS. A simple approach could be to superimpose all objects of a cluster to build the representative as it is depicted in Figure 5.5. However, this approach has the huge drawback that it is only applicable to image data. In addition, the representatives on a higher level of the cluster hierarchy become rather unclear. Therefore, we choose real objects of the data set as cluster representatives.

In the following, CO denotes the cluster ordering (cf. Definition 2.12) from which we want to extract clusters. A cluster $C \subseteq CO$ will be represented by a set of k objects of the cluster, denoted as $\text{REP}(C)$. The number of representatives k can be a user defined number or a number which depends on the size and data distribution of the cluster C.

5.2.1 The Minimum Core Distance Approach

Beside taking medoids, the second approach to choose representative objects of hierarchical clusters uses the density-based clustering notion of OPTICS. The core distance $o.C = \text{CORE}(o)$ of an object $o \in CO$ (cf. Definition 2.10)

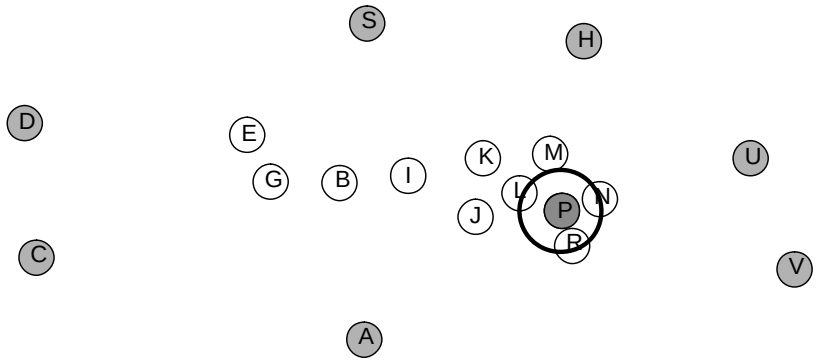


Figure 5.6: Illustration of the minimum core distance approach.

indicates the density of the surrounding region. The smaller the core distance of o, the denser the region surrounding o. This observation lead us to the choice of the object having the minimum core distance as representative of the respective cluster. Formally, $\text{REP}(C)$ can be computed as:

$$\text{REP}(C) := \{o \in C \mid \forall x \in C : o.C \leq x.C\}.$$

We choose the k objects with the minimum core distances of the cluster as representatives. An example is illustrated in Figure 5.6. For $MinPts = 3$ and $k = 1$, we choose object P as representative of the cluster containing the white objects.

The straightforward extension for cluster hierarchies is to choose the k objects from the merged child clusters having the minimum core distances.

Let us note that the choice of the object having the minimum core distance is not determinate because more than k points may have the minimum core distance in a cluster.

5.2.2 The Maximum Successors Approach

The third approach to choose representative objects of hierarchical clusters also uses the density-based clustering notion of OPTICS, but in a more sophisticated way. In fact, it makes use of the density connected relationships between points established by the OPTICS algorithm.

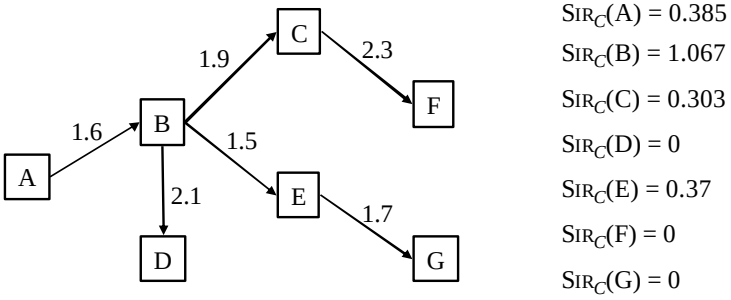


Figure 5.7: Sample successor graph for a cluster of seven objects.

As described in Section 2.3, the result of OPTICS is an ordering of the database, minimizing the reachability distance relation. At each step of the ordering, the object o having the minimum reachability distance w.r.t. the already processed objects before o in the ordering is chosen. Thus, if the reachability distance of object o is not ∞, it is determined by REACHDIST_{den}(p,o) where p is a unique object located before o in the cluster ordering. In Definition 4.3 p is defined as the *predecessor* of o ($\text{PRE}(o) = p$). Based on that concept, the set of *successors* of an object o ($\text{SUC}_{den}(o)$) was formalized in Definition 4.4.

We observed in Section 4.2 that objects may have no predecessor, e.g. each object having a reachability distance of ∞ does not have a predecessor, including the first object in the ordering. On the other hand, some objects may have more than one successor. In that case, some other objects have no successors. An object and its successors need not to be adjacent in the ordering.

We can model the successor-relationship among points within each cluster as a directed *successor graph* where the nodes are the points of one cluster and a directed edge from object o to s represents the relationship $s \in \text{SUC}_{den}(o)$. Each edge (x, y) can further be labeled by $\text{REACHDIST}_{den}(x, y)$ $(= y.R)$. A sample successor graph is illustrated in Figure 5.7.

For the purpose of computing representatives of a cluster, these objects having many successors are interesting. Roughly speaking, these objects are responsible for the most connections within a cluster. The reachability

distance values of these connections further indicate the distance between the objects. In the example cluster visualized in Figure 5.7, object B is responsible for the most connections since its node in the successor graph has the most outgoing edges.

Our third strategy selects the representatives of clusters by maximizing the number of successors and minimizing the according reachability distances. For this purpose, we compute for each object o of a cluster C the Sum of the Invers Reachability distances of the successors of o within C, denoted by $\text{SIR}_C(o)$:

$$\text{SIR}_C(o) := \begin{cases} 0 & \text{if} \quad \text{SUC}_{den}(o) = \emptyset \\ \sum\limits_{\substack{s \in \text{SUC}_{den}(o), \\ s \in C}} \frac{1}{1+\text{REACHDIST}_{den}(o,s)} & \text{otherwise.} \end{cases}$$

We add 1 to REACHDIST_{den}(o,s) in the denominator to weight the impact of the number of successors over the significance of the reachability values. Based on $\text{SIR}_C(o)$, the representatives can be computed as follows:

$$\text{REP}(C) := \{o \in C \mid \forall x \in C : \text{SIR}_C(o) \geq \text{SIR}_C(x)\}.$$

In Figure 5.7, the SIR-values of some objects of the depicted successor graph for a cluster of seven objects are computed. Since D has no successors, $\text{SIR}_C(D)$ is zero. In fact, object B has the highest SIR-value, indicating the central role of B in the cluster: B has three successors with relatively low reachability distance values. Our third strategy selects object B as representative for the cluster.

An illustration of the maximum successor approach is presented in Figure 5.8. For $MinPts = 3$ and $k = 1$, we choose object L as representative of the cluster, containing the white objects (recall that the minimum core distance approach choose P).

Let us note that there is no additional overhead to compute the reachability distances REACHDIST_{den}(o,SUC_{den}(o)) for each $o \in CO$ since these values have been computed by OPTICS during the generation of CO and $\text{REACHDIST}_{den}(o, \text{SUC}_{den}(o)) = \text{SUC}_{den}(o).R$. Furthermore, the result of our

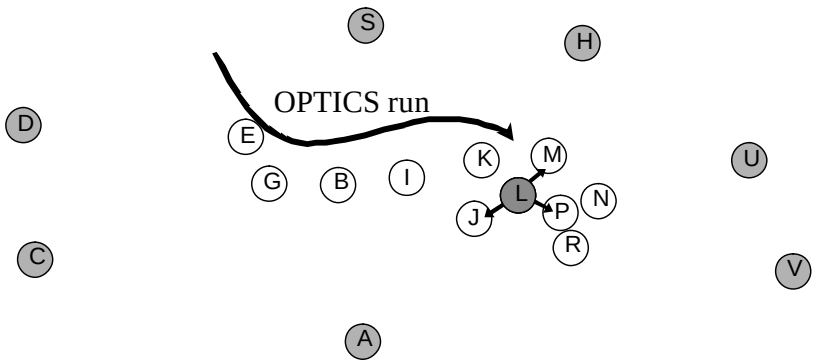


Figure 5.8: Illustration of the maximum successor approach.

selection obviously depends on the order in which the points are processed by OPTICS.

If we want to select k representatives for C, we simply have to choose the k objects with the maximum SIR_C value.

5.2.3 Experimental Evaluation

After a cluster recognition algorithm has analyzed the data, algorithms for cluster representation can help to get a quick visual overview of the data. With the help of representatives, large sets of objects may be characterized through a single object of the data set. We extract sample clusters by applying the GradientClustering algorithm as described in Section 5.1 to the car parts and protein data sets in order to evaluate the different approaches for cluster representatives. In our tests we set the number of representatives to $k = 1$.

The objects of one cluster from the car data set are displayed in Figure 5.9, and the objects of one cluster from the protein data set are displayed in Figure 5.10. The annotated objects are the representatives computed by the respective algorithm. Both, the *Maximum Successor* and the *Minimum Core Distance* approaches give good results. Despite the slight inhomogeneity of the clusters, both representatives sum up the majority of the elements within the clusters. This cannot be said of the representatives computed by the

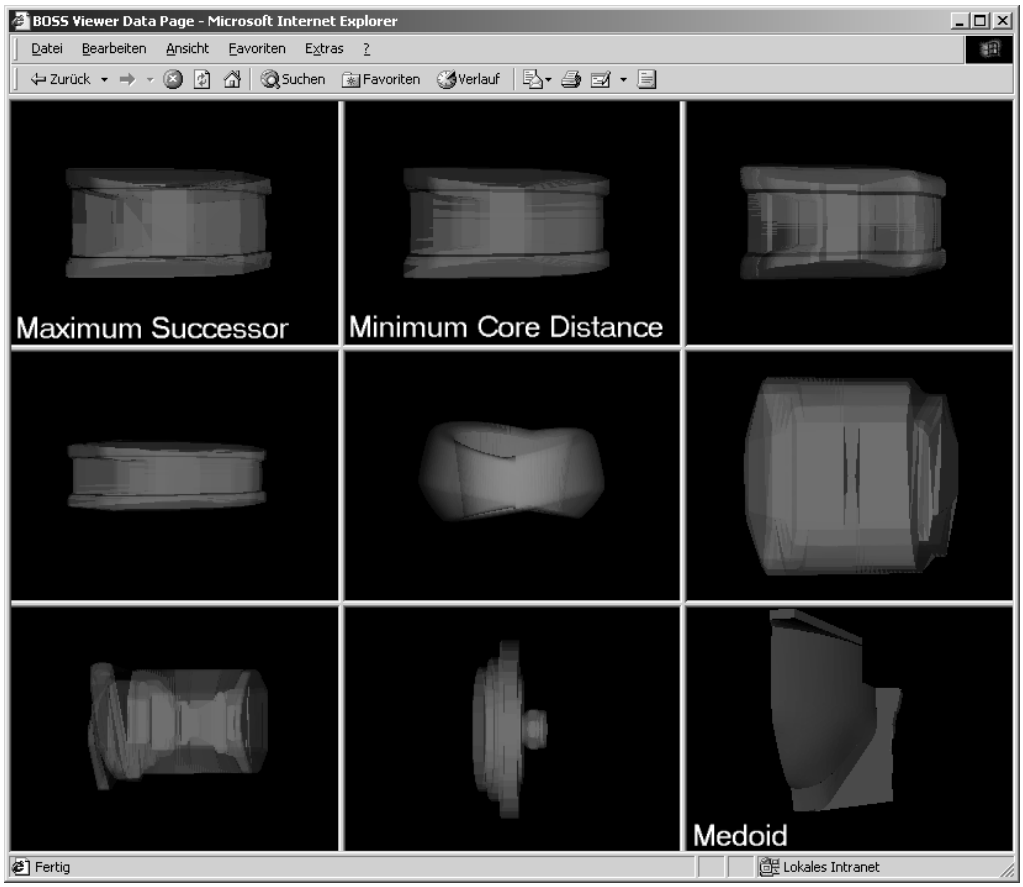


Figure 5.9: A cluster of CAD objects with corresponding representative objects.

commonly used medoid method which selects objects from the trailing end of the cluster. These two clusters and their corresponding representatives are no isolated cases, but reflect our general observations. Nevertheless, there have been some rare cases where the medoid approach yielded the more intuitive representative than the other two approaches. As a consequence, we suggest to use all three approaches within the BOSS system.

If we allow a higher number of representatives, for instance $k = 3$, it might be better to display the representatives of all three approaches to reflect the content of the cluster, instead of displaying the three best representatives of one single approach.

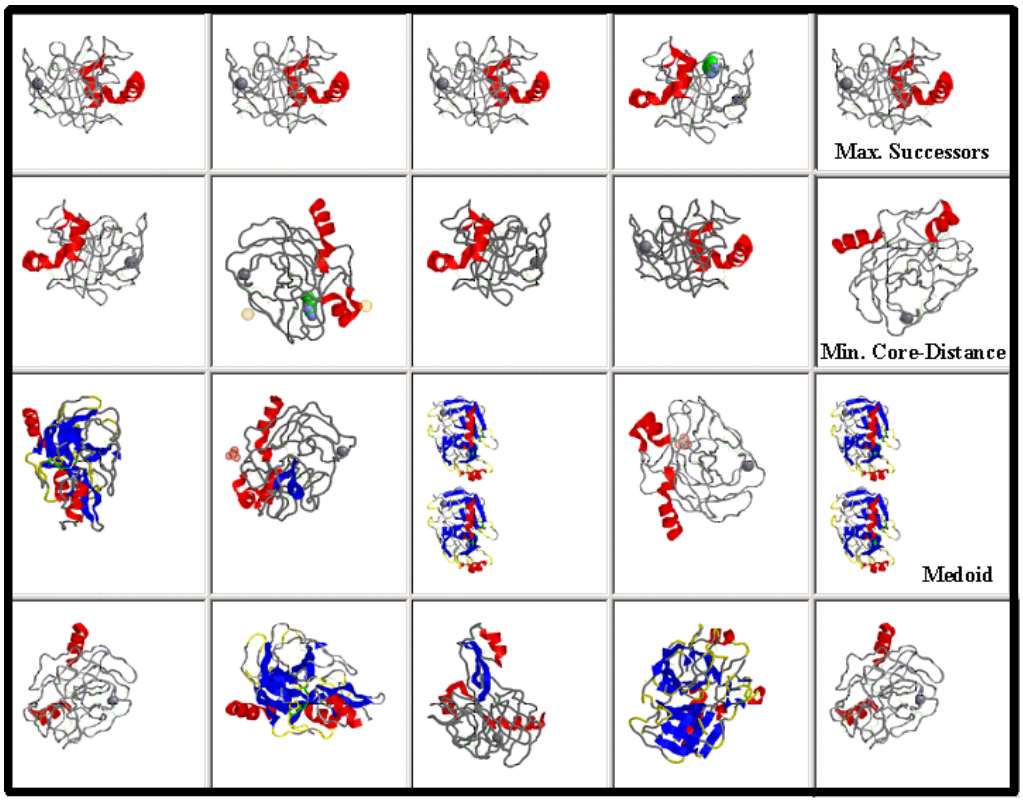


Figure 5.10: A cluster of proteins with corresponding representative objects.

5.3 Summary

Solid cluster extraction from reachability plots and meaningful cluster representation is the heart of our interactive data browsing tool BOSS.

In this chapter, we first proposed a novel approach for cluster recognition that overcomes the problems of existing approaches. For BOSS, it is desirable to find a considerable high number of hierarchically organized clusters to make data browsing sensible. According to this consideration, the proposed cluster extraction algorithm GradientClustering empirically outperforms the recent approaches to cluster extraction. The experimental evaluation confirming this result is based on two real-world data sets, a CAD car data set and a protein structure data set.

Secondly, we investigated two novel approaches to cluster representation in this chapter. We proposed the *Minimum Core Distance* approach and the *Maximum Successor* approach. Both based on the concepts of density connected hierarchical clustering underlying the OPTICS algorithm. Experiments on the car parts database and the protein structure database indicate that both approaches can outperform the simple approach of medoids in terms of intuitive cluster representation. However, we suggest to try always all three approaches of cluster representation. In fact, we integrated all three approaches into the BOSS system.

```
algorithm GradientClustering(ClusterOrdering CO, Integer MinPts, Real t)
  startPts := emptyStack;
  setOfClusters := emptySet;
  currCluster := emptySet;
  o := CO.getFirst();      // first object is a starting point
  startPts.push(o);
  while o.hasNext() do      // for all remaining objects
    o := o.next;
    if o.hasNext() then
      if II(o) > t then      // inflection point
        if gd(o) > 0 then
          if currCluster.size() ≥ MinPts then
            setOfClusters.add(currCluster);
          end if
          currCluster := emptySet;
          if startPts.top().R ≤ o.R then
            startPts.pop();
          end if
          while startPts.top().R < o.R do
            setOfClusters.add(set of objects from startPts.top() to last end point);
            startPts.pop();
          end while
          setOfClusters.add(set of objects from startPts.top() to last end point);
          if o.next.R < o.R then      // o is a starting point
            startPts.push(o);
          end if
        else
          if o.next.R > o.R then      // o is an end point
            currCluster := set of objects from startPts.top() to o;
          end if
        end if
      end if
    else      // add clusters at end of plot
      while not startPts.isEmpty() do
        currCluster := set of objects from startPts.top() to o;
        if (startPts.top().R > o.R) and (currCluster.size() ≥ MinPts) then
          setOfClusters.add(currCluster);
        end if
        startPts.pop();
      end while
    end if
  end while
  return setOfClusters;
```

Figure 5.11: Pseudo code of the GradientClustering algorithm.

Chapter 6

BOSS: Browsing OPTICS Plots for Similarity Search

In this chapter, we describe the application features of the BOSS prototype. Some technical details of the BOSS implementation are discussed in Section 6.1. Some sample applications of BOSS are presented in Section 6.2, including a visual data mining application using protein data and the application of BOSS to evaluate similarity models for voxelized CAD data. A demonstration of the BOSS prototype has been published in [BKK+04]. The chapter concludes with a short summary in Section 6.3.

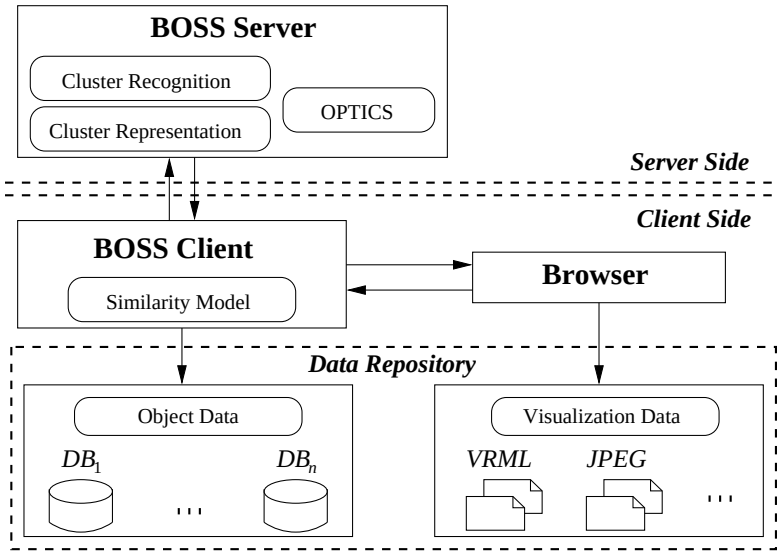


Figure 6.1: BOSS distributed architecture.

6.1 System Architecture

The development of the industrial prototype BOSS is a first step towards developing a comprehensive, scalable and distributed computing solution, designed to make the efficiency of OPTICS and the analytical capabilities of BOSS available to a broader audience. BOSS is implemented as a client/server system allowing users to provide their own data locally along with an appropriate similarity model (cf. Figure 6.1).

The data provided by the user will be comprised of the objects to be clustered as well as a data set to visualize these objects, e.g. VRML files for CAD data (cf. Figure 6.2) or JPEG images for multi-media data. Since this data resides on the user's local computer and is not transmitted to the server, heavy network traffic can be avoided. In order for BOSS to be able to interpret this data, the user must supply his own similarity model with which the reachability data can be calculated. Thus, BOSS can bee also seen as a clustering web service providing all the benefits of BOSS via a web interface.

The independence of the data processing and the data specification enables maximum flexibility. Further flexibility is introduced through the sup-

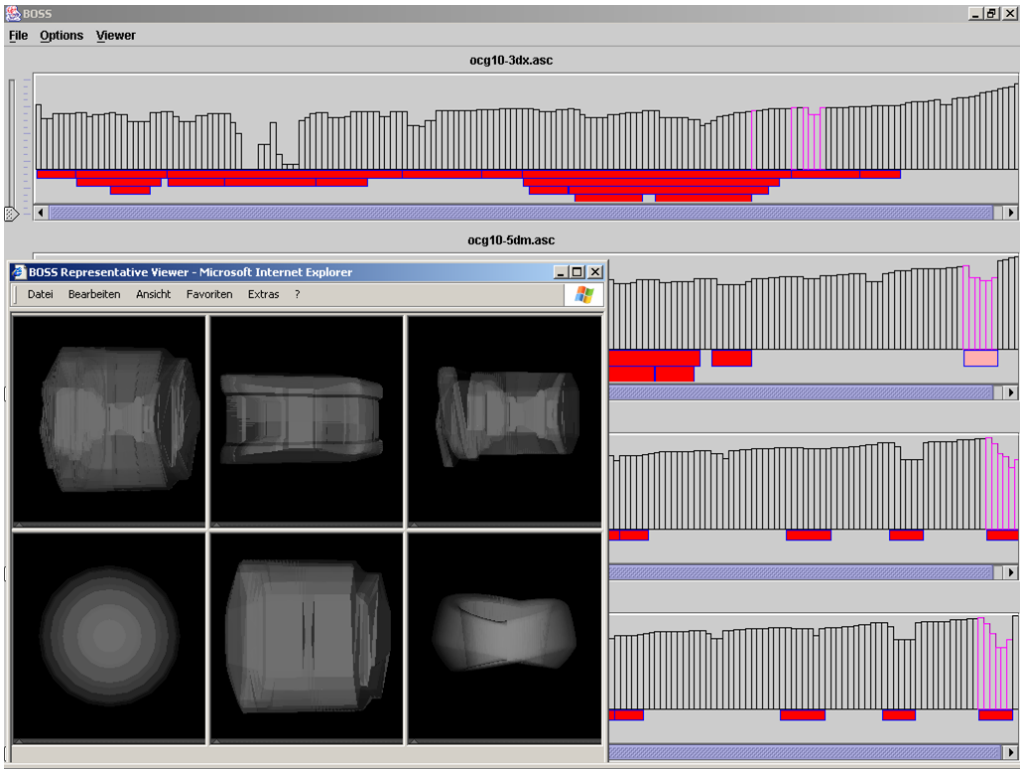


Figure 6.2: BOSS screenshot.

port of external visual representation. As long as the user is capable of displaying the visualization data in a browser, e.g. by means of a suitable plug-in, the browser will then load web pages generated by BOSS, displaying the appropriate data. Thus, multimedia data such as images or VRML files can easily be displayed (cf. Figure 6.2). By externalizing the visualization procedure, we can resort to approved software components which have been specifically developed for displaying objects of the same type as the objects within our clusters.

Figure 6.2 shows a screen shot of the BOSS system where a user evaluates different similarity models. The BOSS application in the background displays four different reachability plots of the same data, generated by using different similarity models. In the foreground, a web browser displays the objects of a cluster of one reachability plot which is marked by the user.

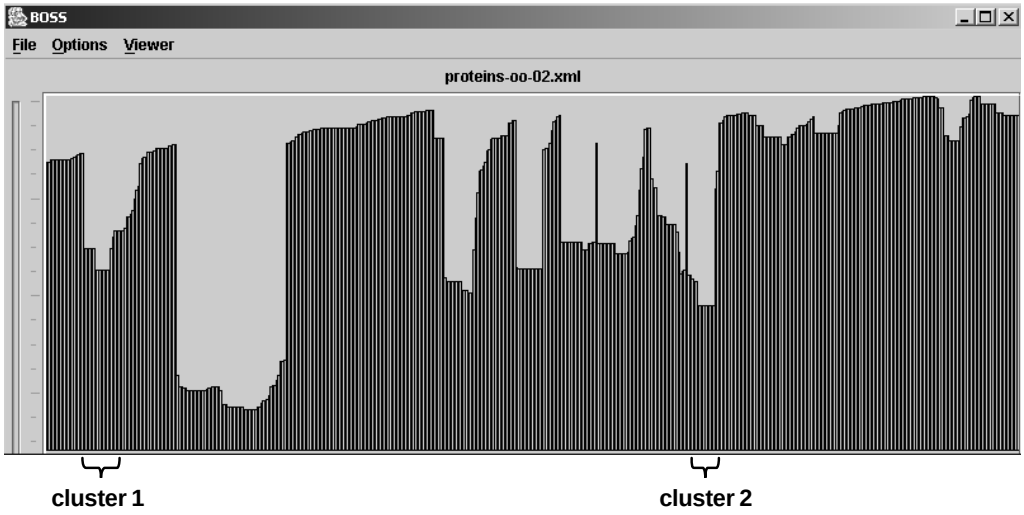


Figure 6.3: OPTICS plot of the protein data set.

6.2 Sample Applications

6.2.1 Visual Data Mining

We applied BOSS to a part of the Protein Data Bank (PDB) [BWF+00] comprising the 3D structural information of about 5,000 proteins (this selection was application specific and done by a domain expert). The protein structures where transformed into feature vectors using the similarity model proposed in [AKKS99, KKS98]. A biological expert used BOSS for visually mining through the underlying part of the PDB, in particular, to visualize the resulting clusters for semi-automatic cluster analysis. The resulting reachability plot generated by OPTICS is visualized in Figure 6.3. We used GIF images to visualize the proteins. Some resulting clusters are depicted in Figure 6.4 (cluster 1) and Figure 6.5 (cluster 2). Based on the cluster analysis, several new and interesting insights into the PDB where gained. For example, cluster 1 exhibits several structural similarities among proteins that were previously unknown to our user. Cluster 2 contains variants of the same protein which is combined with different ligands in each data entry. In fact, the information of cluster 2 enabled our user to prune the redundant variants of this protein for further analysis. With the help of BOSS, the

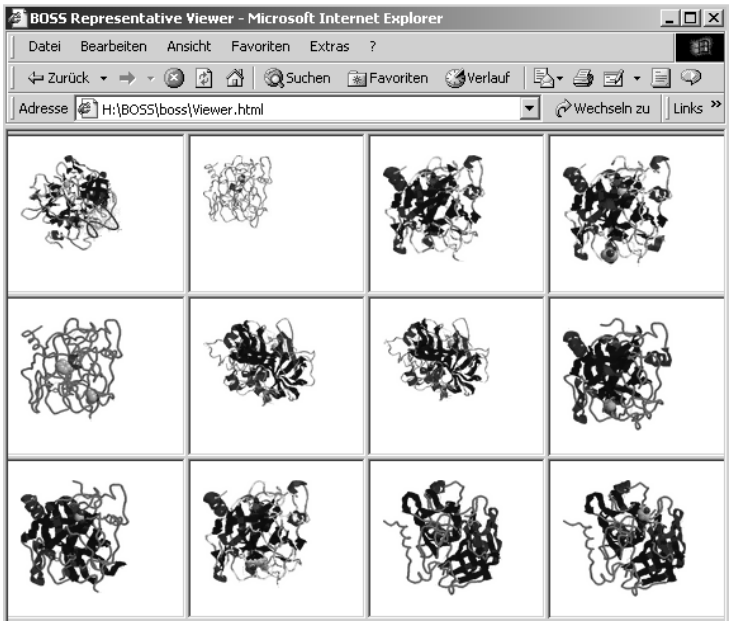


Figure 6.4: Sample cluster 1 found on the protein database.

analysis of the clustering results was rather user-friendly. The big advantage of BOSS is its applicability to a broad variety of data sets as long as the data objects comprise a suitable visualization that can be handled by a standard browser. In fact, the semi-automatic cluster analysis is dramatically simplified by using BOSS.

6.2.2 Evaluation of Similarity Models

We also applied BOSS to a CAD database containing voxelized car parts in order to evaluate several similarity models for voxelized CAD data. We compared the three space partitioning models presented in [KKM+03] and the object partitioning model presented in [KBK+03].

The space partitioning models are based on a partitioning of the data space into buckets or cells. In our case, since the data objects are voxelized (i.e. are represented as a set of voxels), the three-dimensional data space was partitioned by using an axis-parallel grid of fixed cell width. Each cell of the three-dimensional grid corresponds to one or more attributes of the resulting

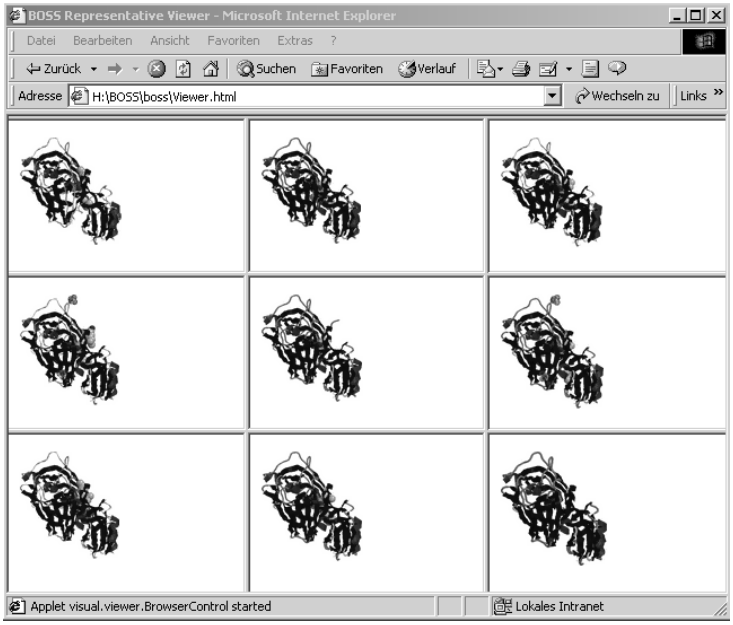
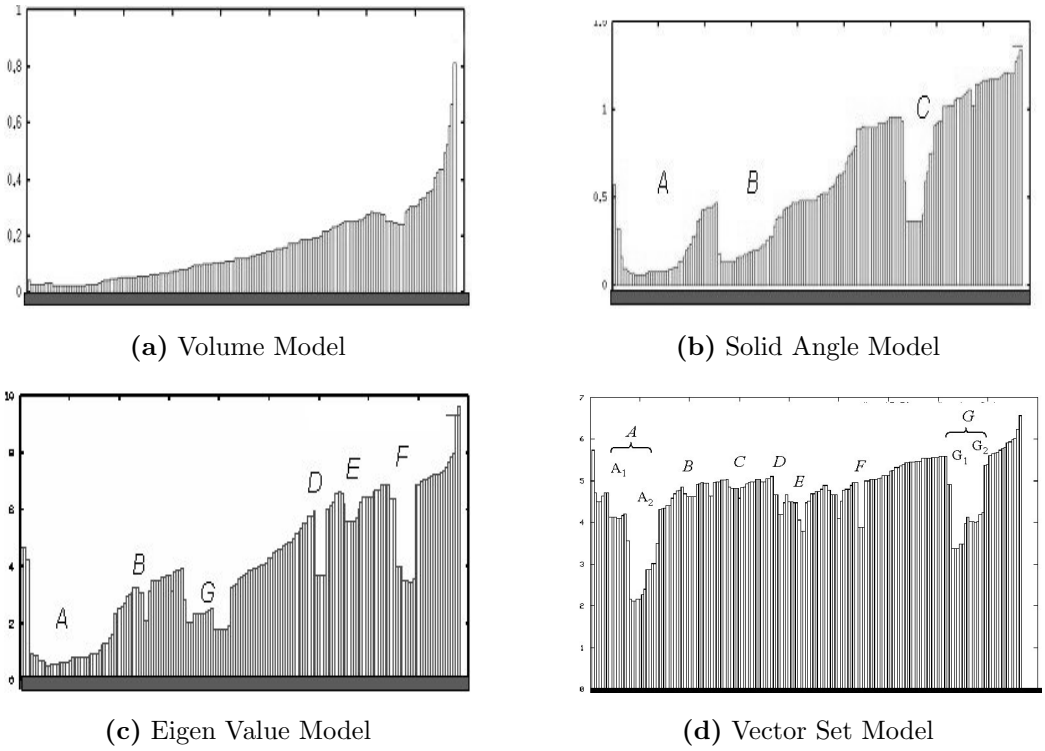


Figure 6.5: Sample cluster 2 found on the protein database.

feature vectors that represent the voxelized parts. The models differ in the kind of features that are extracted from each cell:

- The Volume Model extracts the proportion of the object's volume in each cell as feature (i.e. the number of object voxels normalized by the total number voxels in each cell).
- The Solid Angle Model extracts the mean value of Solid-Angle values of the object's surface voxels in each cell as feature (the value is 0 if no voxel is in the according cell and 1 if the according cell contains voxels of an object but no surface voxels). The Solid Angle value [Con86] is a measurement for the convexity or concavity of surfaces.
- The Eigen Value Model extracts the three eigen values of the object voxels in each cell as features. The resulting feature vector contains three times more attributes than grid cells.

The object partitioning model is based on a decomposition of the object by means of covers [Jag91]. Each object is transformed in a sequence of cover-



(a) Volume Model

(b) Solid Angle Model

(c) Eigen Value Model

(d) Vector Set Model

Figure 6.6: Reachability plots computed by OPTICS using different similarity models.

segments that cover the object perfectly. The Vector Set Model [KBK+03] extracts several features for each cover-segment of one object. Since the data objects are composed of several such cover-segments, they are represented as a set of feature vectors, each representing a single cover-segment. A suitable distance function on sets of feature vectors is defined in [KBK+03] for similarity search purposes.

Using BOSS, the evaluation of these four models turned out to be rather easy because the clustering power of each model is revealed at a glance and BOSS allows a comparative analysis of the resulting cluster hierarchy. The four reachability plots are visualized in Figure 6.6. The contents of some sample clusters in the plots are depicted in Figure 6.7. As it can be seen in Figure 6.6(a), the Volume Model is rather unsuitable for the car data set. The according clustering structure computed by OPTICS does not reveal

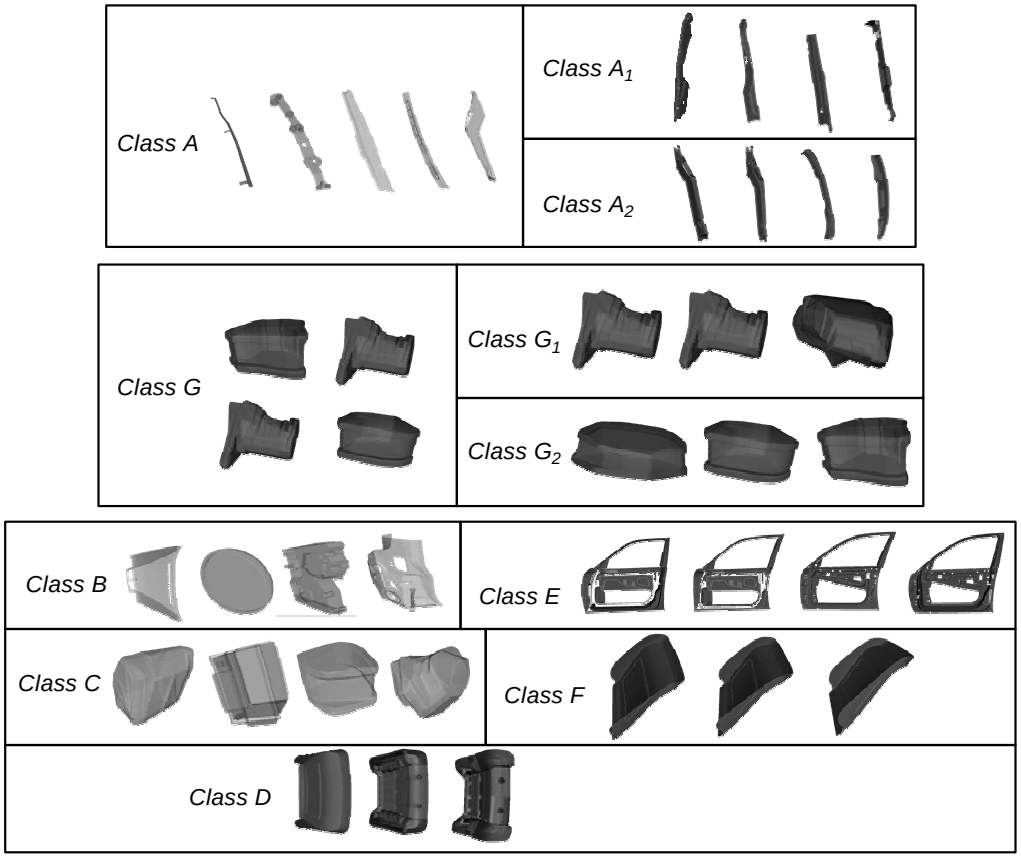


Figure 6.7: Contents of the clusters detected in Figure 6.6.

any clusters. Slightly better results are achieved by the Solid Angle Model (cf. Figure 6.6(b)). At least, clusters A, B, and C can be detected. However, when we browsed the clusters with BOSS, it turned out that all three clusters contained additional parts that are not intuitively similar. On the other hand, the Eigen Value Model reflects the intuitive notion of similarity rather well. Many classes of car parts are detected, even the hierarchy of the objects in cluster G split in sub-clusters G_1 and G_2 is detected (cf. Figure 6.6(c)). Finally, the Vector Set Model reflects the intuitive notion of similarity best. Using this model, OPTICS detects the most clusters (cf. Figure 6.6(d)) and the most hierarchical structure. However, the difference between the Eigen Value Model and the Vector Set Model is only marginal.

In summary, the evaluation of the similarity models using OPTICS is

rather objective and is extremely simplified by BOSS.

6.3 Summary and Discussion

In this chapter, we introduced some details regarding the implementation of BOSS. In addition, we outlined two sample applications of BOSS. The first is an application to visual data mining. We used BOSS for semi-automatic cluster analysis of a database of protein structures. BOSS significantly simplifies this procedure of extracting valuable knowledge. The second application of BOSS was to evaluate different similarity models for voxelized CAD data. We compared four different models using a database of car parts. Again, BOSS significantly simplifies this evaluation and allows the deduction of important hints for the usability of each model. The evaluation of the models using hierarchical clustering is much more objective than applying sample k-nn queries because all data objects are taken into account for the evaluation rather than some sample (random) data objects.

Part III

Adopting Density-Based Clustering to High Dimensional Data

Chapter 7

Clustering High Dimensional Data

Clustering high dimensional data is usually a difficult task. In fact, most traditional ("full dimensional") clustering algorithms tend to break down when applied to high dimensional feature spaces. The reasons for this behavior is also known by the term *curse of dimensionality* and are worked out within this chapter in Section 7.1. Since the importance of clustering high dimensional data is steadily increasing with new data generation capabilities, new approaches have been developed recently to address this problem. Section 7.2 provides a general classification of these approaches. Section 7.3 outlines two motivating examples and describes some data sets used as an evaluating test bed for the methods proposed in the next chapters.

7.1 The Curse of Dimensionality

In this section, we will explore some general properties of high dimensional feature spaces that have an impact on the performance of clustering algorithms. These phenomena are usually summed up by the term *curse of dimensionality*. Let us note that there are several properties contributing to the curse of dimensionality that may be missed in this section, but are less important in the context of this thesis.

Observation 7.1 *The probability that points are located at the border of the data space increases with growing dimensionality.*

The correctness of this observation can be made clear with the following considerations. If we assume uniform distribution of the data points inside a hypercube with side length 1, i.e. $\mathcal{D} \subseteq [0,1]^d$ (cf. Figure 7.1 left), the volume of such a data space is $1^d = 1$. The probability $P_{\text{surface}}(r)$ that a point randomly taken from a uniform and independent distribution in a d-dimensional space has a distance of r or below to the space boundary can be determined as given below:

$$P_{\text{surface}}(r) = 1 - (1 - 2 \cdot r)^d.$$

As it is shown in Figure 7.1 (right), the probability that a point is inside a 10% border of the data boundary rapidly increases with growing dimensionality. For $d = 3$ dimensions, $P_{\text{surface}}(0.1)$ is already 0.488% and reaches 0.965% for $d = 15$ dimensions.

Observation 7.2 *In high dimensional feature spaces, the ε-neighborhoods of the points will most likely exceed the boundaries of the data space.*

Due to Observation 7.1, the points tend to be located nearer to the boundaries of the data space with increasing dimensionality. As a consequence the hypersphere of the ε-range query of these points, growing with each dimension, will exceed the boundaries of the data space. Since density-based clustering

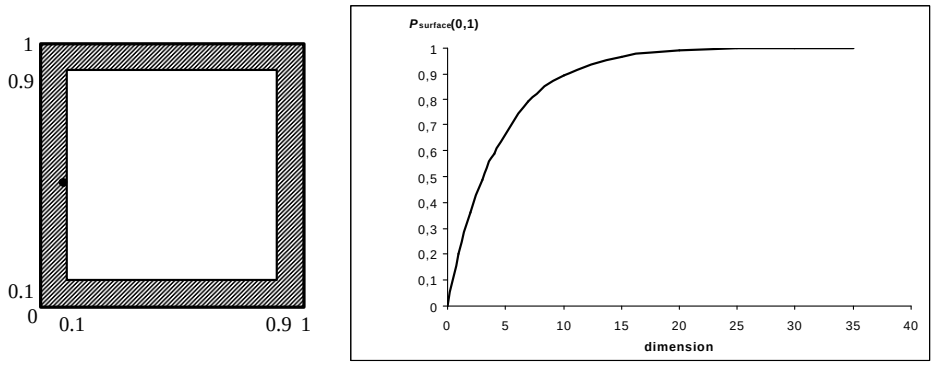


Figure 7.1: Probability of a point near by the data space boundary.

works on top of ε-neighborhoods, this observation may cause problems. If the points are located at the boundary of the data space, the ε-neighborhood of these points are usually "smaller" because they exceed the boundaries of the data space, i.e. the probability that they contain a certain number of points decreases.

The first two observations have an impact on the density-based clustering notion. However, the next observation challenges the entire idea of clustering in high dimensional feature spaces.

Observation 7.3 *In high dimensional feature spaces, the furthest neighbor of a point is usually as far as the nearest neighbor.*

In [HAK00] the authors experimentally show that with growing dimensionality the concept of density tends to become meaningless because nearest and furthest neighbors of objects tend to be no more discriminable. Concepts like nearest neighbor or ε-neighborhood also tend to become meaningless in high dimensional spaces. The general consequence of this observation is that clustering makes no sense in high dimensional feature spaces because the data objects usually do not cluster any more but are sparsely distributed.

Most clustering methods mentioned in Section 2.1 compute "full dimensional" clusters in a given feature space, i.e. each dimension of this feature space is equally weighted when computing the distance between points. These approaches are successful for low-dimensional feature spaces. However, in higher dimensional feature spaces, their accuracy and/or efficiency deteriorates significantly due to the curse of dimensionality; in particular due to Observation 7.3.

7.2 General Approaches for Clustering High Dimensional Data

As we have seen in the previous section, clustering high dimensional data is usually a hopeless task because in high dimensional feature spaces, the data objects do not cluster anymore. In addition, many features may be irrelevant and/or strongly correlated. Nevertheless, clustering such high dimensional data is mandatory in many applications. Thus, novel clustering approaches especially developed for high dimensional data are necessary.

A common approach to cope with high dimensional feature spaces in many contexts including clustering is the application of a dimensionality reduction technique before clustering. Dimensionality reduction techniques such as Principal Component Analysis (PCA) map all objects of the data set onto a particular subspace while minimizing the loss of information. A standard clustering method can then be used to compute clusters in this subspace. However, if different subsets of the objects cluster well on different subspaces of the feature space, a dimensionality reduction will most likely fail. An example is visualized in Figure 7.2: two subsets of a 3-dimensional data set are projected onto two different 2-dimensional subspaces. One subset can be clustered well when projected onto the subspace spanned by the x-/y-axes, whereas it is scattered significantly along the z-axis. The second subset clusters well in the projection on the subspace spanned by the y-/z-axes and scatters significantly along the x-axis. The application of a global dimensionality reduction method on this sample data set would yield a rather

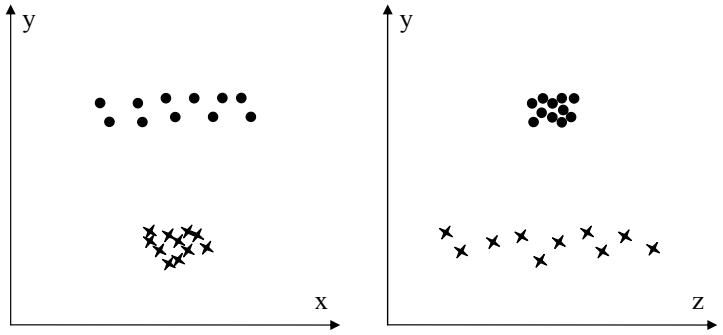


Figure 7.2: Sample projected clusters in different subspaces.

high loss of information or may not yield an appropriate, significantly lower dimensional subspace.

In general, global feature reduction techniques cannot be applied when different subsets of features are irrelevant in different subgroups of data objects. As a consequence, in recent years several original approaches have been investigated to solve the problem of clustering high dimensional data. These approaches try to incooperate the task of feature selection within the clustering procedure. In the following, we give a classification of these approaches. The basis of these approaches is that points in a high dimensional feature space usually cluster in (different) subspaces of this feature space.

Projected clustering. The goal of projected clustering is to compute a flat partition of the data into k *projected clusters*, i.e. to assign a unique cluster-ID to each data point. Intuitively, a projected cluster is a set of points in a high dimensional feature space, having a low variance in one or more (but not all) attributes, and a higher but arbitrary variance in the remaining attributes. Projected clustering methods map each cluster to its associated subspace, allowing more flexibility than global methods projecting the entire data set onto a single subspace. The subspace of a particular cluster may in general vary significantly from the subspaces of the other clusters. Objects not belonging to any projected cluster should be classified as noise. Figure 7.2 illustrates two projected clusters in a sample 3-dimensional space spanned by attributes $\{x, y, z\}$. One subset of points (indicated by dots) cluster in the

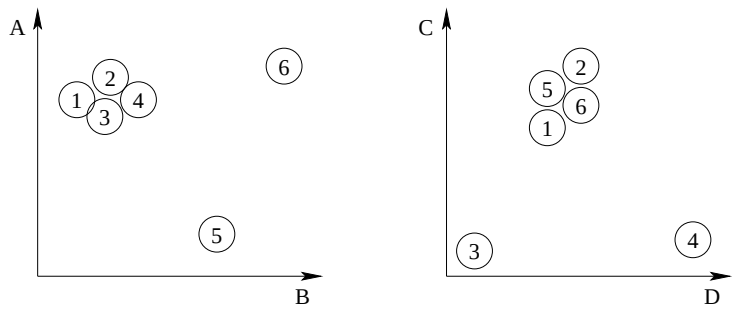


Figure 7.3: Sample objects cluster differently in varying subspaces.

projection onto subspace $\{y, z\}$ whereas the rest of the points cluster in the projection onto subspace $\{x, y\}$. In Section 9.5.2, we will propose a density-based approach for the projected clustering problem as a specialization of correlation clustering (see below).

Subspace clustering. While the projected clustering approach is more flexible than dimensionality reduction, it suffers from the fact that the information of objects which are clustered differently in varying subspaces is lost. Figure 7.3 illustrates this problem using a feature space of four attributes A, B, C, and D. In the subspace {AB} the objects 1 and 2 cluster together with objects 3 and 4, whereas in the subspace {CD} they cluster with objects 5 and 6. Either the information of the cluster in subspace {AB} or in subspace {CD} will be lost. In recent years, the task of *subspace clustering* was introduced to overcome these problems. Subspace clustering is the task of automatically detecting clusters in subspaces of the original feature space. In Chapter 8, we introduce density-based approaches to the subspace clustering problem.

Pattern-based clustering. Projected clustering algorithms and subspace clustering algorithms search for dense regions in subsets of the entire feature space. The similarity between points is measured using the distance of the points in the according subspace. Thus, these approaches are sometimes also called *distance-based* methods. However, a more general kind of similarity can also be rather interesting in several applications. Using such a more gen-

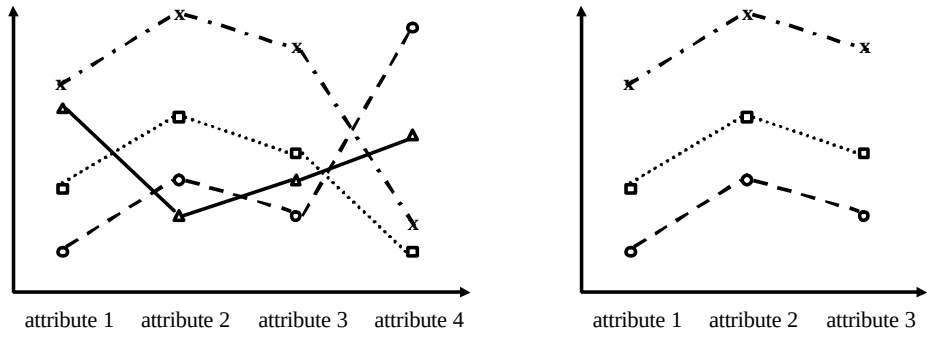
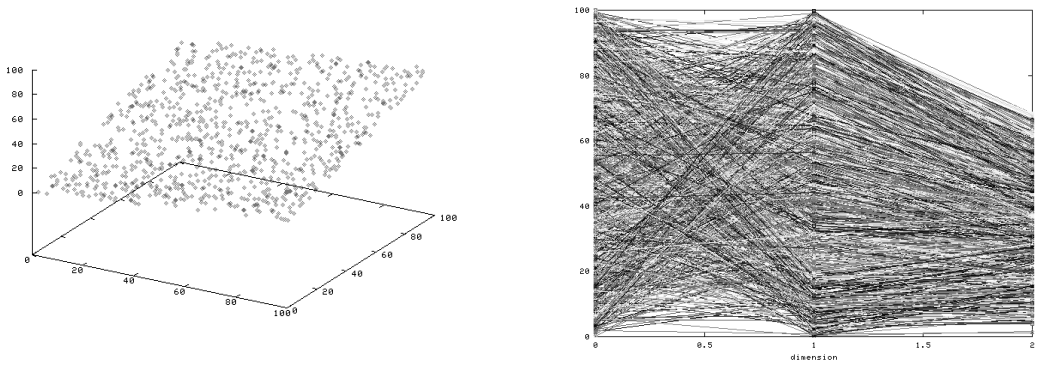


Figure 7.4: Transposed view (left) and pattern-based cluster (right) of some sample database objects.

eral similarity notion, *pattern-based clustering* algorithms search for groups of points that exhibit a similar tendency (or pattern) in a subset of their attributes. An example similar pattern could be a common shift of the attribute values. The absolute attribute values need not to be similar, so the resulting clusters need not to be dense in the according subspace. However, subspace clusters that exhibit a certain density can be seen as a special case of a pattern-based cluster. A common shift can be easily visualized by a so-called *transposed view* of the data points, i.e. a 2-dimensional view where the attributes are plotted along the x-axis, and the value of each attribute is plotted along the y-axis. Then, pattern-based clusters can be visually seen as points that exhibit a common pattern in a subset of their attributes. An example is illustrated in Figure 7.4. The transposed view of four 4-dimensional points are depicted on the left. Three of these objects form a pattern-based cluster because they exhibit a common pattern in the first three attributes (cf. Figure 7.4(right)).

Correlation clustering. Correlation clustering is a mixture of distance-based approaches and pattern-based methods. Projected clustering algorithms and subspace clustering algorithms are usually not able to capture local data correlations and find clusters of correlated objects. The principal axes of correlated data are arbitrarily oriented. In contrast, projected and subspace clustering techniques only find axis-parallel projections of the



(a) 3-dimensional view **(b)** Transposed view

Figure 7.5: A 2-dimensional correlation plane in a 3-dimensional feature space.

data. Similar limitations hold for pattern-based algorithms that detect only positive linear correlations but cannot detect negative correlations nor correlations where one attribute is determined by two or more attributes. Figure 7.5 illustrates such an example where the 3-dimensional data points exhibit a 2-dimensional linear correlation (which can be seen as a 2-dimensional hyperplane in Figure 7.5(a)). In particular, two attributes are independent, whereas the third attribute is a linear combination of the first two attributes. The transposed view of this set does not exhibit a common pattern (cf. Figure 7.5(b)). Finding such sets of points that exhibit both, density and arbitrary linear correlation, is the task of *correlation clustering*. In Chapter 9, we introduce a density-based correlation clustering approach.

7.3 Sample Applications

We will focus on two applications of clustering high dimensional data which are described in the following. We will evaluate the methods that will be proposed in the following using data sets from these applications. Let us note that both applications come from molecular biology and are of great practical impact.

7.3.1 Gene Expression Analysis

Proteins are the building blocks of cells in living organisms. It is mandatory for cells to produce identical copies of proteins in a large amount. The blueprints to produce such identical copies are coded in the genes. The protein production is a very complex procedure, so we will give only a short overview of the mechanism. First, the gene that codes for the requested protein is "read" and transcribed into an intermediate, called messenger RNA (mRNA). This procedure is called *gene expression*. In a second step, parts of this mRNA are translated into a copy of the desired protein or may have other functionalities within the organism. Roughly speaking, the so-called expression level of a gene is a measurement for the frequency the gene is expressed, i.e. transcribed into its mRNA product. Thus, the expression level of a gene allows conclusions about the current amount of the protein in a cell the gene codes for.

Micro-array chip technologies enable biologists to measure the expression level of thousands of genes simultaneously under different conditions or in different tissues. Usually, gene expression data appears as a matrix where the rows represent genes, and the columns represent samples (e.g. different experiments, time slots, test persons, etc.). The value of the i-th feature of a particular gene is the expression level of this gene in the i-th sample (cf. Figure 7.6 left). Figure 7.6 (right) shows a visualization of a raw data matrix from a real gene expression experiment.

It is interesting from a biological point of view to cluster both the rows (genes) and the columns (samples) of the matrix, depending on the research scope. Clustering the genes is the method of choice if one searches for co-expressed genes, i.e. genes, whose expression levels are similar, and co-regulations between genes, i.e. linear dependencies between the expression levels of genes. Co-expression and co-regulation usually indicates that the genes are functionally related. Throughout this thesis, we will focus on clustering genes to find co-expressions and co-regulations between genes.

When clustering the genes to detect co-expressed genes, one has to cope with the problem that usually the co-expression of the genes can only be

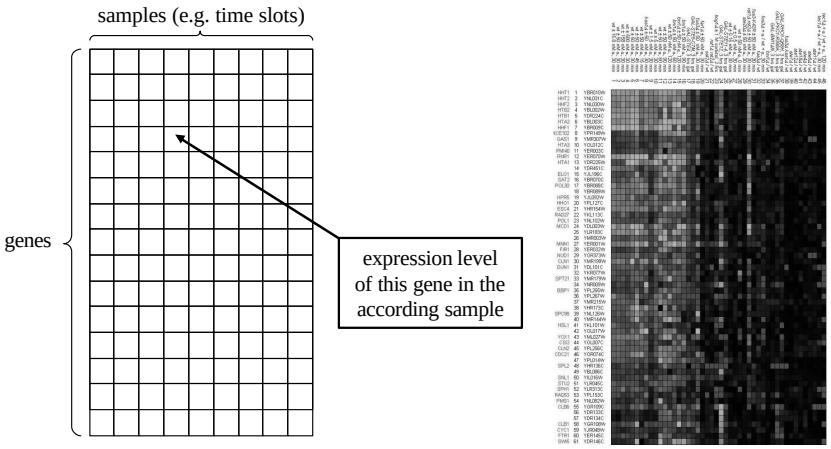


Figure 7.6: Gene expression data matrix: schematic view (left), visualization of a sample raw data excerpt (right).

detected in subsets of the samples. Moreover, genes may have several different functions that are needed in different subsets of the samples. In other words, different subsets of the attributes (samples) are responsible for different co-expressions of the genes. Let us note that when clustering the samples to identify e.g. homogeneous groups of patients, this situation is even worse. As various phenotypes (e.g. hair color, gender, cancer type, etc.) are hidden in varying subsets of the genes, the samples could usually be clustered differently according to these phenotypes, i.e. in varying subspaces.

As a consequence, from the biological point of view, it is interesting to apply a projected clustering algorithm to find co-expressed genes. In addition, it is even more interesting to apply a subspace clustering method to gene expression data. The information of varying co-expressions of genes under different conditions or in different tissues is necessary to understand the regulation of gene expression and the interaction of the encoded proteins which are both key information for many industrial applications such as drug design. Last but not least, correlation clustering could be applied to find linear dependencies of genes (i.e. co-regulations between genes). The information of co-regulation is also very important to understand the regulation of gene expression.

Table 7.1: Summarization of gene expression data sets.

Data set	Number of genes (n)	Number of samples (d)	Reference
Spellman	4381	24	[SSZ+98]
Tavazoie	2884	17	[THC+99]

In this thesis, we will validate the proposed methods using two benchmark gene expression data sets. Both data sets study the mitotic cell cycle of yeast, a reference organism for molecular biology, and are derived from a time series experiment.

The first data set (CDC15 mutant set from [SSZ+98]), in the following referred to as "Spellman" data, is suitable to detect co-expressions. Thus, it was used to validate the subspace clustering algorithms proposed in Chapter 8. It contains the expression level of 6,000 genes measured at 24 different time slots. Since some genes have missing expression values and the handling of missing values in gene expression analysis is a non-trivial task, we eliminated those genes from our test data set. The resulting data set contains around 4,400 genes expressed at 24 different time slots.

The second data set [THC+99], in the following referred to as "Tavazoie" data, is suitable for detecting co-regulations. Thus it was used to validate the correlation clustering algorithm. The expression levels of approximately 3,000 genes are measured at 17 different time slots.

Table 7.1 summerizes the features of both gene expression sets used for evaluation in this thesis.

The resulting clusters are evaluated in terms of functional relationships of the genes in the same cluster. A cluster is regarded as biologically meaningful if it contains a significant amount of functionally related genes. We tested the genes according to the following three biologically proven criteria, indicating that two genes are functionally related:

1. known direct interactions of the genes or the according gene products,

2. known common complexes of the genes or the according gene products,

3. participation of the according gene products in common pathways.

We used the publicly available Saccharomyces Genome Database (SGD)[Sac] for this analysis.

Let us note that the validation of the clustering results on gene expression data sets is sometimes delicate. The data generation procedure is rather error-prone. Thus, gene expression data is usually very noisy, and may lead to several meaningless clusters. Nevertheless, the detection of homogeneous clusters containing functionally related genes is significant and confirms the accuracy of the according clustering method. Since both data sets contain several genes of yet unknown functions, cluster analysis has also some predictive power for the functions of these genes.

Both test data sets were clustered using OPTICS in the full-dimensional space. The resulting clusters did not contain a significant amount of functional related genes.

7.3.2 Metabolic Screening of Newborns

The metabolome is the entirety of metabolites in a cell or organism. Substances that are imported into cells or organisms (e.g. food) are usually converted through a tremendous cascade of biochemical reactions. Consecutive sequences of such reactions are called metabolic pathways that are directed and controlled by special proteins called enzymes. Genetic diseases may cause the presence or absence of particular enzymes and thus, some undesired reactions may take place or some necessary reactions do not. The resulting ("metabolic") diseases may heavily affect the organisms.

Biologists and medical researchers try to investigate these diseases by so-called metabolic screenings. A metabolic screening measures the concentration of dedicated metabolites in the blood of patients. Unfortunately, the information about the relationships of metabolism concentration and disease is rather small. However, pioneering projects like the Bavarian metabolic newborn screening [LNRvK+02] are first steps to get a data set of sufficient

Table 7.2: Class distribution of the Metabolome data set.

Class	Number of newborns
control	1.400
LCHAD	60
MCAD	53
PKU	306
others	181

information of the relationships of metabolite concentration and metabolic diseases. Data mining methods such as cluster analysis are a key step to extract useful information and thus to gain new insights for medical treatment of metabolic diseases. In particular, the application of a correlation clustering algorithm is important to find correlations between the concentrations of the metabolites, significant for a special disease. In addition, correlation clustering is needed in the future to find new groups of newborns suffering from unknown diseases. This would provide important information for the analysis of the huge metabolic screening data collected by the newborn screening program in Bavaria, Germany [LNRvK+02].

In this thesis, we will validate the proposed correlation clustering method using a small part of the metabolome screening data set of [LNRvK+02], in the following referred to as "Metabolome" data set. It measures the concentrations of 43 metabolites in the blood of 2,000 human newborns. The newborns were labeled according to some specific metabolic diseases such as PKU. The distribution of classes are summarized in Table 7.2. The healthy newborns are labeled with "control". The aim of the cluster analysis is to distinguish the predefined classes of this benchmark data set in a best possible way.

7.4 Summary

In this section, we outlined the need of original clustering approaches for high dimensional data. First, we discussed some phenomena occurring in

high dimensional feature spaces which are known by the term curse of dimensionality. We observed, that points usually do not cluster anymore when the data dimensionality increases but tend to cluster in subspaces of the original feature space. After that, we classified the clustering approaches designed especially for high dimensional data. In particular, we identified four classes of methods: projected clustering, subspace clustering, pattern-based clustering, and correlation clustering. Last but not least, we described two sample real-world applications where these new approaches play a central role in data analysis: gene expression analysis and metabolic screening. Both applications demand the clustering of high dimensional feature data that usually cannot be worked-out by traditional (Full-dimensional) clustering algorithms. We also introduced two gene expression data sets and one metabolic screening data set that will serve as benchmark data sets used in the evaluation of the methods that will be presented in the consecutive chapters.

Chapter 8

Subspace Clustering

Subspace clustering is getting increasing attention from the research community. In this chapter, we propose a density-based solution to the subspace clustering problem. First, we review and discuss recent subspace clustering algorithms in Section 8.1. Then, we investigate the foundations of density-based subspace clustering in Section 8.2, presenting an extension of the (traditional) density-based concepts to subspace clustering. Based on these concepts, the density-based subspace clustering algorithm *SUBCLU* and its semi-hierarchical extension *RIS* is introduced in Section 8.3 and Section 8.4, respectively. The concepts presented in this chapter are major extensions of the material published in [KKKW03] and [KKK04]. The chapter is concluded by a short summary in Section 8.5.

8.1 Related Work

Recent work has been done to tackle the problem of subspace clustering. In the following, current approaches are reviewed with no claim on completeness.

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

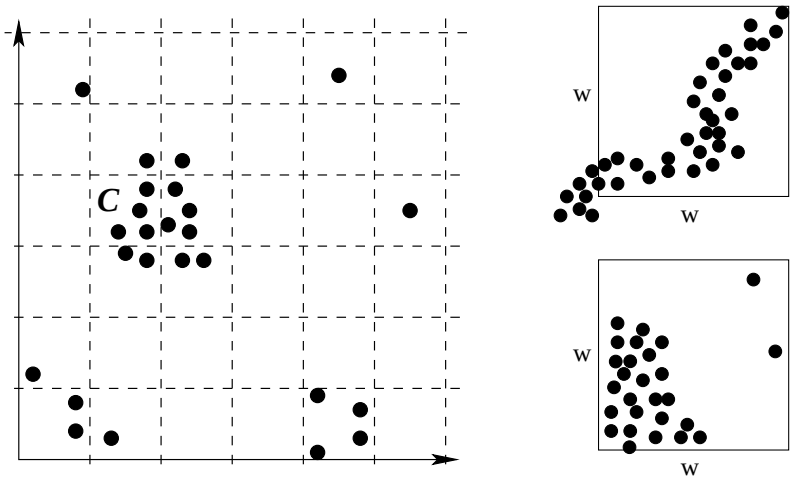
A slight modification of CLIQUE is the algorithm ENCLUS (ENtropy-based CLUStering) [CFZ99]. The major difference is the criterion used for subspace selection. The criterion of ENCLUS is based on entropy computation of a discrete random variable. The entropy of any subspace S is high

when the points are uniformly distributed in S whereas it is lower the more closely the points in S are packed. Subspaces with an entropy below an input threshold ω are considered as suitable for clustering. A monotonicity criterion is presented, enabling the use of a similar bottom-up algorithm as in CLIQUE [CFZ99].

A more significant modification of CLIQUE is presented in [GNC99] and [NGC01], introducing the algorithm MAFIA (Merging of Adaptive Finite IntervAls). MAFIA uses adaptive, variable-sized grids in each dimension. A dedicated technique based on histograms which aims at merging grid cells is used to reduce the number of bins compared to CLIQUE. An input parameter α is used as a so-called *cluster dominance factor* to select bins which are α-times more densely populated (relative to their volume) than the average. The algorithm starts to produce such one-dimensional dense units as candidates and proceeds recursively in higher dimensions. In contrast to CLIQUE, MAFIA uses any two k-dimensional dense units to construct a new $(k+1)$-dimensional candidate as soon as they share an arbitrary $(k-1)$-face (not only first dimensions). As a consequence, the number of generated candidates is much larger compared to CLIQUE. Neighboring dense units are merged to form clusters. Redundant clusters, i.e. clusters that are true subsets of higher dimensional clusters, are removed.

A big drawback of all these methods is caused by the use of grids. In general, grid-based approaches heavily depend on the positioning of the grids. Figure 8.1(a) illustrates this problem for CLIQUE: Each grid by itself is not dense, if $\tau > 4$, and thus, the cluster C is not found. On the other hand, if $\tau = 4$, at least a part cluster C is found but the cell with four objects in the lower right corner just above the x-axis is reported as a cluster. In fact, we have to set $\tau = 1$ in order to detect the complete cluster C. Entire clusters or parts of it may also be missed if they are inadequately oriented or shaped.

In [PJAM02] a mathematical definition of an "optimal projected cluster" is presented along with a Monte Carlo algorithm called DOC (Density-based Optimal projective Clustering) to compute an approximation of such an optimal projected cluster. Using the user-specified input parameters w and α, an



(a) CLIQUE **(b)** DOC

Figure 8.1: Illustration of drawbacks of existing subspace clustering algorithms.

optimal projected cluster is defined as a set of points $C \subseteq \mathcal{D}$ associated with a subspace of dimensions $S \subseteq \mathcal{A}$ such that C is α-dense (i.e. contains more than $\alpha\%$ points of the database) and the projection of C onto the subspace spanned by S must be contained in a hyper-cube of width w. In all other dimensions $a_i \notin S$ the points in C are not contained in a hyper-cube of width w. The proposed algorithm DOC takes a random seed point and computes an optimal set of dimensions S for this seed from a randomly taken subset of $\mathcal{D}$. All points that are within distance w in the projection onto S from the seed point are included into the cluster C. The algorithm has to be applied multiple times to find several subspace clusters.

DOC only finds approximations of subspace clusters, because it generates projected clusters of width $2w$. In addition, no assumption on the distribution of points inside C and thus inside the hyper-cube of width w in the subspace spanned by S is made. The reported projected clusters may contain additional noise objects, especially when the size of the projected cluster is considerably smaller than $2w$. Furthermore, the reported clusters may miss some points that naturally belong to the projected cluster, especially when the size of the projected cluster is considerably larger than $2w$.

Both problems are illustrated in Figure 8.1(b). The runtime of DOC applied to a d-dimensional data set of n points for finding a single "optimal" projected cluster is $O(n \cdot d^c)$ where c is a constant, depending on the two input parameters [PJAM02].

8.2 Foundations of Density-Based Subspace Clustering

In this section, we formalize the notion of density-based subspace clustering by adapting the density-based clustering concepts to the subspace clustering problem. After that, we will explore monotonicity properties for these concepts.

8.2.1 Adapting Density-Based Concepts to Subspace Clustering

The concepts of density-based clustering as described in full details in Section 2.3 are defined for "full-dimensional" clustering. In the following, we adopt these definitions to the problem of subspace clustering in order to develop a density-based subspace clustering method.

We start with the basic concept of a projected ε-neighborhood.

Definition 8.1 (ε-neighborhood in a subspace)
Let $\varepsilon \in \mathbb{R}$, $S \subseteq \mathcal{A}$ and $o \in \mathcal{D}$. The ε-neighborhood *of o in S, denoted by $\mathcal{N}_\varepsilon^S(o)$, is defined by*

$$\mathcal{N}_\varepsilon^S(o) = \{x \in \mathcal{D} \mid dist(\pi_S(o), \pi_S(x)) \leq \varepsilon\}.$$

Based on the definition of ε-neighborhood in a subspace, the core point property in a subspace is straightforward.

Definition 8.2 (core point in a subspace)
A point $o \in \mathcal{D}$ is a core point in a subspace $S \subseteq \mathcal{A}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$, *denoted by* $\textsc{Core}^S_{den}(o)$, *if its ε-neighborhood in S contains at least $MinPts$ objects, formally:*

$$\textsc{Core}^S_{den}(o) \Leftrightarrow |\mathcal{N}^S_\varepsilon(o)| \geq MinPts.$$

Let us note that the acronym *den* in the definition refers to 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.

Direct density reachability in a subspace can then be defined as follows.

Definition 8.3 (direct density reachable in a subspace)
A point $p \in \mathcal{D}$ is directly density reachable *from $q \in \mathcal{D}$* in a subspace $S \subseteq \mathcal{A}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$, *denoted by* $\textsc{DirReach}^S_{den}(q, p)$, *if q is a core point in S and p is an element of $\mathcal{N}^S_\varepsilon(q)$, formally:*

$$\textsc{DirReach}^S_{den}(q, p) \Leftrightarrow \textsc{Core}^S_{den}(q) \ \wedge \ p \in \mathcal{N}^S_\varepsilon(q).$$

Direct density reachability in a subspace is obviously symmetric only for pairs of core points in a subspace.

Definition 8.4 (density reachable in a subspace)
A point $p \in \mathcal{D}$ is density reachable *from $q \in \mathcal{D}$* in a subspace $S \subseteq \mathcal{A}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$, *denoted by* $\textsc{Reach}^S_{den}(q, p)$, *if there is a chain of points $p_1, \ldots, p_n \in \mathcal{D}$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density reachable from p_i, formally:*

$$\begin{aligned}
&\textsc{Reach}^S_{den}(q, p) \Leftrightarrow \\
&\exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = q \ \wedge \ p_n = p \ \wedge \\
&\forall i \in \{1, \ldots, n-1\} : \textsc{DirReach}^S_{den}(p_i, p_{i+1}).
\end{aligned}$$

Density reachability in a subspace is obviously the transitive enclosure of direct density reachability in a subspace, but it is still only symmetric for a pair of core points in a subspace.

Definition 8.5 (density connected in a subspace)
A point $p \in \mathcal{D}$ is density connected *to a point $q \in \mathcal{D}$* in a subspace $S \subseteq \mathcal{A}$ *w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$, denoted by $\textsc{Connect}^S_{den}(q,p)$, if there is a point o such that both p and q are density reachable from o, formally:*

$$\textsc{Connect}^S_{den}(q,p) \Leftrightarrow$$
$$\exists o \in \mathcal{D} : \textsc{Reach}^S_{den}(o,q) \wedge \textsc{Reach}^S_{den}(o,p).$$

Density connected in a subspace is symmetric in general.

Definition 8.6 (density connected set in a subspace)
A non-empty subset $C \subseteq \mathcal{D}$ is called a density connected set in a subspace *$S \subseteq \mathcal{A}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$, denoted by $\textsc{ConSet}^S_{den}(C)$ if all objects in C are density connected in S, formally:*

$$\textsc{ConSet}^S_{den}(C) \Leftrightarrow \forall p,q \in C : \textsc{Connect}^S_{den}(p,q).$$

Density connected clusters in a subspace can then be defined as given below.

Definition 8.7 (density connected cluster in a subspace)
A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ is called a density connected cluster in a subspace *$S \subseteq \mathcal{A}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$, denoted by $\textsc{Cluster}^S_{den}(\mathcal{C})$, if $\mathcal{C}$ is a density connected set in S and $\mathcal{C}$ is maximal w.r.t. density-reachability in S, formally:*

$\textsc{Cluster}^S_{den}(\mathcal{C}) \Leftrightarrow$

(1) *Connectivity:* $\textsc{ConSet}^S_{den}(\mathcal{C})$

(2) *Maximality:* $\forall p, q \in \mathcal{D} : q \in \mathcal{C} \wedge \textsc{Reach}^S_{den}(q, p) \Rightarrow p \in \mathcal{C}$.

A flat density-based decomposition of a database in a subspace is defined as follows.

Definition 8.8 (flat density-based decomposition in a subspace)
A flat density-based decomposition *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$ *of* $\mathcal{D}$ *in a subspace* $S \subseteq \mathcal{A}$ *is a decomposition* D^S_{den} *of* $\mathcal{D}$ *into* $k \geq 1$ *subsets such that* $k-1$ *subsets are density connected sets in* S *and the* k*-th (possible empty) set contains the noise points, formally:*

$$D^S_{den} = \{C_1, \ldots, C_{k-1}, N\} \qquad \textit{where}$$
$$\neg\, \textsc{Cluster}^S_{den}(N) \wedge \forall i \in \{1, \ldots, k-1\} : C_i \neq \emptyset \wedge \textsc{Cluster}^S_{den}(C_i).$$

Let us note that by adopting the full-dimensional density-based clustering notion, our subspace clustering notion is able to detect subspace clusters of different size and shape. In addition, we can obviously compute a density decomposition in a subspace S by applying DBSCAN to the projection of $\mathcal{D}$ onto S.

8.2.2 Monotonicity Properties

To use an efficient strategy for the search through all possible subspaces, we have to examine monotonicity properties of density-based clusters. We start with core points in a subspace.

Lemma 8.1 (monotonicity of core points)
Let $\varepsilon \in \mathbb{R}$, $MinPts \in \mathbb{N}$, *and* $S \subseteq \mathcal{A}$. *If* $o \in \mathcal{D}$ *is a core point in subspace* S, o *is also a core point in any projection* $T \subset S$, *formally:*

$$\forall T \subset S : \textsc{Core}^S_{den}(o) \Rightarrow \textsc{Core}^T_{den}(o).$$

Proof.

$$
\begin{aligned}
&\textsc{Core}^S_{den}(o)\\
&\quad\Leftrightarrow |\,\mathcal{N}^S_\varepsilon(o)\,| \geq \mathit{MinPts}\\
&\quad\Leftrightarrow |\{x \,|\, \mathit{dist}(\pi_S(o), \pi_S(x)) \leq \varepsilon\}| \geq \mathit{MinPts}\\
&\quad\Leftrightarrow |\{x \,|\, \sqrt[p]{\sum_{i=1}^{\mathit{dim}[S]} (\pi_{\{a_i\}}(o) - \pi_{\{a_i\}}(x))^p} \leq \varepsilon\}| \geq \mathit{MinPts}\\
&\quad\overset{(T\subseteq S)}{\Rightarrow} |\{x \,|\, \sqrt[p]{\sum_{i=1}^{\mathit{dim}[T]} (\pi_{\{a_i\}}(o) - \pi_{\{a_i\}}(x))^p} \leq \varepsilon\}| \geq \mathit{MinPts}\\
&\quad\Leftrightarrow |\{x \,|\, \mathit{dist}(\pi_T(o), \pi_T(x)) \leq \varepsilon\}| \geq \mathit{MinPts}\\
&\quad\Leftrightarrow |\,\mathcal{N}^T_\varepsilon(o)\,| \geq \mathit{MinPts}\\
&\quad\Leftrightarrow \textsc{Core}^T_{den}(o).
\end{aligned}
$$

□

Based on the monotonicity of core points, we can examine the monotonicity of direct density reachable.

Lemma 8.2 (monotonicity of direct density reachable)
Let $\varepsilon \in \mathbb{R}$, $\mathit{MinPts} \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. If a point $q \in \mathcal{D}$ is direct density reachable from a point $o \in \mathcal{D}$ in S, q is also direct density reachable from o in any projection $T \subset S$, formally:

$$\forall T \subseteq S : \textsc{DirReach}^S_{den}(o, q) \Rightarrow \textsc{DirReach}^T_{den}(o, q).$$

Proof.

$$
\begin{aligned}
&\textsc{DirReach}^S_{den}(o, q)\\
&\quad\Leftrightarrow \textsc{Core}^S_{den}(o) \wedge q \in \mathcal{N}^S_\varepsilon(o)\\
&\quad\Leftrightarrow \textsc{Core}^S_{den}(o) \wedge \mathit{dist}(\pi_S(o), \pi_S(q)) \leq \varepsilon\\
&\quad\Leftrightarrow \textsc{Core}^S_{den}(o) \wedge \sqrt[p]{\sum_{i=1}^{\mathit{dim}[S]} (\pi_{\{a_i\}}(o) - \pi_{\{a_i\}}(q))^p} \leq \varepsilon
\end{aligned}
$$

$$\overset{(T\subseteq S),\, Lemma\ 8.1}{\Longrightarrow} \text{CORE}^T_{den}(o) \wedge \sqrt[p]{\sum_{i=1}^{dim[T]} (\pi_{\{a_i\}}(o) - \pi_{a_i}(q))^p} \leq \varepsilon$$

$$\Leftrightarrow \text{CORE}^T_{den}(o) \wedge dist(\pi_T(o), \pi_T(q)) \leq \varepsilon$$

$$\Leftrightarrow \text{CORE}^T_{den}(o) \wedge q \in \mathcal{N}^T_\varepsilon(o)$$

$$\Leftrightarrow \text{DIRREACH}^T_{den}(o, q).$$

□

Based on the monotonicity of direct density reachable, we can examine the monotonicity of density reachable.

Lemma 8.3 (monotonicity of density reachable)
Let $\varepsilon \in \mathbb{R}$, $MinPts \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. If a point $q \in \mathcal{D}$ is density reachable from a point $o \in \mathcal{D}$ in S, q is also density reachable from o in any projection $T \subset S$, formally:

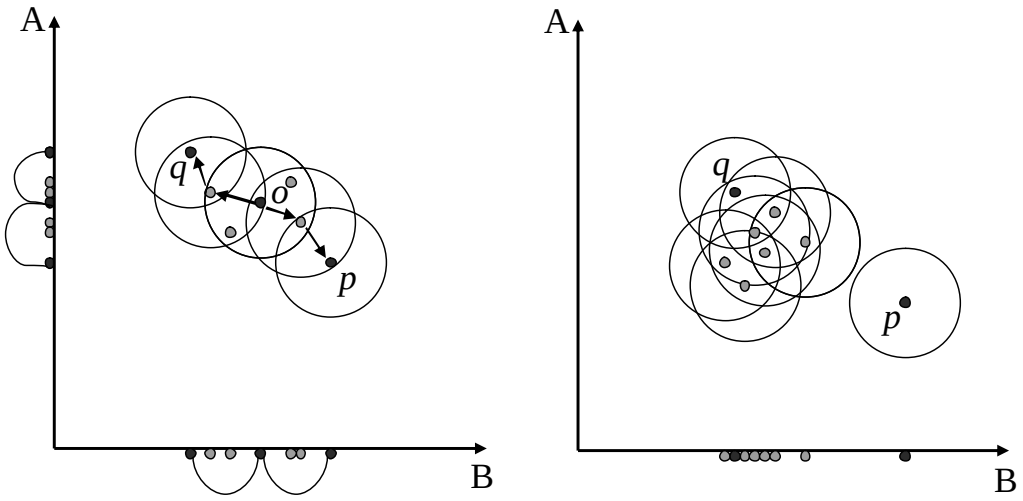
$$\forall T \subseteq S : \text{REACH}^S_{den}(o, q) \Rightarrow \text{REACH}^T_{den}(o, q).$$

Proof.

$$\begin{aligned}
&\text{REACH}^S_{den}(o, q) \\
&\quad \Leftrightarrow \exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = o \wedge p_n = q \wedge \forall i \in \{1, \ldots, n-1\} : \\
&\qquad \text{DIRREACH}^S_{den}(p_i, p_{i+1}) \\
&\quad \overset{(T\subseteq S),\, Lemma\ 8.2}{\Longrightarrow} \exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = o \wedge p_n = q \wedge \\
&\qquad\qquad \forall i \in \{1, \ldots, n-1\} : \text{DIRREACH}^T_{den}(p_i, p_{i+1}) \\
&\quad \Leftrightarrow \text{REACH}^T_{den}(o, q).
\end{aligned}$$

□

Based on the monotonicity of density reachable, we can prove the monotonicity of density connected.



(a) p and q are density connected by o **(b)** p and q are not density connected

Figure 8.2: Monotonicity of density connected (the circles indicate the ε-neighborhoods, $MinPts = 4$).

Lemma 8.4 (monotonicity of density connected)
Let $\varepsilon \in \mathbb{R}$, $MinPts \in \mathbb{N}$, and $S \subseteq \mathcal{A}$. If a point $q \in \mathcal{D}$ is density connected to a point $o \in \mathcal{D}$ in S, q is also density connected to o in any projection $T \subset S$, formally:

$$\forall T \subseteq S : \textsc{Connect}_{den}^{S}(o, q) \Rightarrow \textsc{Connect}_{den}^{T}(o, q).$$

Proof.

$$\begin{aligned}
&\textsc{Connect}_{den}^{S}(o, q) \\
&\quad \Leftrightarrow \exists x \in \mathcal{D} : \textsc{Reach}_{den}^{S}(x, o) \wedge \textsc{Reach}_{den}^{S}(x, q) \\
&\quad \overset{(T \subseteq S),\ Lemma\ 8.3}{\Longrightarrow} \exists x \in \mathcal{D} : \\
&\qquad \textsc{Reach}_{den}^{T}(x, o) \wedge \textsc{Reach}_{den}^{T}(x, q) \\
&\quad \Leftrightarrow \textsc{Connect}_{den}^{T}(o, q).
\end{aligned}$$

□

The monotonicity of density connected is illustrated in Figure 8.2. In Figure 8.2(a), p and q are density connected via o in the subspace spanned

by attributes A and B, i.e. subspace $\{A, B\}$. Thus, p and q are also density connected via o in each subspace $\{A\}$ and $\{B\}$ of $\{AB\}$. The inverse conclusion is depicted in Figure 8.2(b): p and q are not density connected in subspace $\{B\}$. Thus, they are also not density connected in the superspace $\{AB\}$, although they might be density connected in subspace $\{A\}$.

Lemma 8.5 (monotonicity of density connected sets)
Let $\varepsilon \in \mathbb{R}$, MinPts $\in \mathbb{N}$, and $S \subseteq \mathcal{A}$. If a non-empty subset $C \subseteq \mathcal{D}$ is a density connected set in S, it is also a density connected set in any projection $T \subset S$, formally:

$$\forall T \subseteq S : \textsc{ConSet}_{den}^{S}(C) \Rightarrow \textsc{ConSet}_{den}^{T}(C).$$

Proof.

$$\begin{aligned}
&\textsc{ConSet}_{den}^{S}(C) \\
&\quad \Leftrightarrow \forall o, q \in C : \textsc{Connect}_{den}^{S}(o, q) \\
&\quad \overset{(T \subseteq S),\ Lemma\ 8.4}{\Longrightarrow} \forall o, q \in C : \textsc{Connect}_{den}^{T}(o, q) \\
&\quad \Leftrightarrow \textsc{ConSet}_{den}^{T}(C).
\end{aligned}$$

□

Although density connected sets are monotonic, density connected clusters are not monotonic in general as we will see in the next section. Nevertheless, the monotonicity properties presented in this section provide a solid basis for an efficient subspace clustering algorithm.

8.3 Density-Based Subspace Clustering

8.3.1 General Idea

A straightforward approach for density-based subspace clustering would be to run DBSCAN in all possible subspaces to detect all density connected

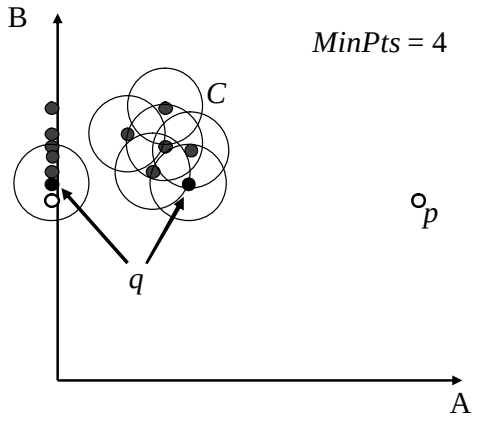


Figure 8.3: Visualization of a density connected cluster C loosing its maximality w.r.t. density reachability in a subspace.

clusters. The problem is that the number of subspaces is $O(2^d)$. A more effective strategy would be to use the clustering information of previous subspaces in the process of generating all clusters and drop all subspaces that cannot contain any density connected clusters.

Unfortunately, density connected clusters are not monotonic, i.e. if $C \subseteq \mathcal{D}$ is a density connected cluster in subspace $S \subseteq \mathcal{A}$, it need not to be a density connected cluster in any $T \subseteq S$. The reason for this is that in T the density connected cluster C needs not to be maximal w.r.t. density reachability any more. There may be additional points which are not in C but are density-reachable in T from a point in C. Figure 8.3 illustrates this fact on a simple 2-dimensional example. The density connected set C (containing point q) in the 2-dimensional subspace spanned by attributes A and B (containing the points painted in black) obviously does not contain point p (painted in white) but is maximal w.r.t. density reachability, i.e. C is a density connected cluster in subspace $S = \{A, B\}$. On the other hand, in the projection onto subspace $T = \{A\} \subseteq \{A, B\}$, point p is obviously density reachable from q, and thus belongs to the same cluster. As a consequence, the density connected set C is not maximal w.r.t. density reachability any more, i.e. it is not a cluster for itself but a part of a cluster in T.

However, density connected sets are monotonic (cf. Lemma 8.5). In fact,

if $C \subseteq \mathcal{D}$ is a density connected set in subspace $S \subseteq \mathcal{A}$, then C is also a density connected set in any subspace $T \subseteq S$.

The inversion of Lemma 8.5 is the key idea for an efficient bottom-up algorithm to detect the density connected sets in all subspaces of high dimensional data. Due to this inversion, we do not have to examine any subspace S if at least one $T_i \subset S$ contains no cluster (i.e. a density connected set). On the other hand, Figure 8.3 not only visualizes that clusters are not monotonic but also gives us a hint what can happen with a density connected cluster when we add a dimension: some points (in this example p) may disappear, or (not illustrated in Figure 8.3) a cluster may be split in several subclusters. Thus, if all $T_i \subset S$ contain clusters, we have to test whether the clusters in all $T_i \subset S$ are conserved in S. However, all points that are noise in T_i are also noise in S. As a consequence, we do not need to consider noise objects in T_i when we generate the clusters in S.

8.3.2 Algorithm SUBCLU

SUBCLU is based on a bottom-up, greedy algorithm to detect the density-connected clusters in all subspaces of high dimensional data. The algorithm is presented in Figure 8.4. We use the procedure $DBSCAN(\mathcal{D}, S, \varepsilon, MinPts)$ to compute a flat density connected decomposition of $\mathcal{D}$ in subspace S w.r.t. ε and $MinPts$. In addition, the following data structures are used (cf. Figure 8.4):

- $\mathcal{C}^S$ denotes the set of all density connected clusters of $\mathcal{D}$ in subspace S (w.r.t. ε and $MinPts$) and can be computed by the procedure $DBSCAN(\mathcal{D}, S, \varepsilon, MinPts)$ (the input parameters ε and $MinPts$ are fixed), i.e. $\mathcal{C}^S := DBSCAN(\mathcal{D}, S, \varepsilon, MinPts)$. Let us note, that $\mathcal{C}^S$ is the density-based decomposition of $\mathcal{D}$ in S (D^S_{den}) without the noise set.

- $\mathcal{S}_k$ denotes the set of all k-dimensional subspaces containing at least one cluster, i.e. $\mathcal{S}_k := \{S \mid dim[S] = k \wedge \mathcal{C}^S \neq \emptyset\}$.

algorithm SUBCLU(SetOfObjects $\mathcal{D}$, Real ε, Integer $MinPts$)
/* STEP 1 Generate all 1-D clusters */
  $\mathcal{S}_1 := \emptyset$;
  $\mathcal{C}_1 := \emptyset$;
  for each $a_i \in \mathcal{A}$ **do**
    $\mathcal{C}^{\{a_i\}} := DBSCAN(\mathcal{D}, \{a_i\}, \varepsilon, MinPts)$;
    if $\mathcal{C}^{\{a_i\}} \neq \emptyset$ **then**
      $\mathcal{S}_1 := \mathcal{S}_1 \cup \{a_i\}$;
      $\mathcal{C}_1 := \mathcal{C}_1 \cup \mathcal{C}^{\{a_i\}}$;
    end if
  end for
/* STEP 2 Generate $(k+1)$**-D clusters from** k**-D clusters */**
  $k := 1$;
  while $\mathcal{S}_k \neq \emptyset$ **do**
    $CandS_{k+1}$:= GenerateCandidateSubspaces(S_k);
    for each $cand \in CandS_{k+1}$ **do**
      $bestSubspace := \min_{T \in \mathcal{S}_k \wedge T \subset cand} \sum_{C_i \in \mathcal{C}^T} |C_i|$
      $\mathcal{C}^{cand} := \emptyset$;
      for each cluster $cl \in \mathcal{C}^{bestSubspace}$ **do**
        $\mathcal{C}^{cand} = \mathcal{C}^{cand} \cup DBSCAN(cl, cand, \varepsilon, MinPts)$;
        if $\mathcal{C}^{cand} \neq \emptyset$ **then**
          $\mathcal{S}_{k+1} := \mathcal{S}_{k+1} \cup cand$;
          $\mathcal{C}_{k+1} := \mathcal{C}_{k+1} \cup \mathcal{C}^{cand}$;
        end if
      end for
    end for
    $k := k + 1$
  end while

Figure 8.4: The SUBCLU algorithm.

- $\mathcal{C}_k$ denotes the set of sets of all clusters in k-dimensional subspaces, i.e. $\mathcal{C}_k := \{\mathcal{C}^S \mid dim[S] = k\}$.

We begin with generating all 1-dimensional clusters by applying DBSCAN to each 1-dimensional subspace (STEP 1 in Figure 8.4).

For each detected cluster we have to check whether this cluster is (or parts of it are) still existent in higher dimensional subspaces. Due to Lemma 8.5 no other clusters can exist in higher dimensional subspaces. Thus, for each k-dimensional subspace $S \in \mathcal{S}_k$, we search all other k-dimensional subspaces $T \in \mathcal{S}_k$ $(T \neq S)$ having $(k-1)$ attributes in common and join them to generate $(k+1)$-dimensional candidate subspaces (STEP 1 of the procedure

```
method GenerateCandidates(SetOfSubspaces S_k)
  CandS_{k+1} := ∅;
/* STEP 1 Join */
  for each s_1 ∈ S_k do
    for each s_2 ∈ S_k do
      if s_1.attr_1 = s_2.attr_1 ∧ ... ∧ s_1.attr_{k-1} = s_2.attr_{k-1} ∧ s_1.attr_k < s_2.attr_k
       then
        insert {s_1.attr_1, ..., s_1.attr_k, s_2.attr_k} into CandS_{k+1};
      end if
    end for
  end for
/* STEP 2 Prune */
  for each cand ∈ CandS_{k+1} do
    for each s ⊂ cand with |s| = k do
      if s ∉ S_k then
        delete cand from CandS_{k+1};
      end if
    end for
  end for
```

Figure 8.5: Procedure GenerateCandidates.

GenerateCandidates in Figure 8.5). The set of $(k+1)$-dimensional candidate subspaces is denoted by $CandS_{k+1}$.

Due to Lemma 8.5, for each candidate subspace $S \in CandS_{k+1}$, $\mathcal{S}_k$ must contain each k-dimensional subspace $T \subset S$ $(dim[T] = k)$. We can prune these candidates having at least one k-dimensional subspace not included in $\mathcal{S}_k$ (STEP 2 of procedure GenerateCandidates in Figure 8.5). This reduces the number of $(k+1)$-dimensional candidate subspaces.

In the last step (STEP 2 in Figure 8.4), we generate the $(k+1)$-dimensional clusters and the corresponding $(k+1)$-dimensional subspaces containing these clusters, using the k-dimensional subclusters and the list of $(k+1)$-dimensional candidate subspaces. For that purpose, we have to do the following: for each candidate subspace $cand \in CandS_{k+1}$ we take one k-dimensional subspace $T \subset cand$ and simply call the above mentioned procedure $DBSCAN(cl, cand, \varepsilon, m)$ for each cluster cl in T $(cl \in \mathcal{C}^T)$ to generate $\mathcal{C}^{cand}$. To minimize the cost of the run of DBSCAN in each $cand$, we choose that subspace $bestSubspace \subset cand$ from $\mathcal{S}_k$ in which a minimum number of

objects are in the cluster, i.e.

$$bestSubspace := \text{argmin}_{T \in \mathcal{S}_k \wedge T \subset cand} \sum_{C_i \in \mathcal{C}^T} |C_i|.$$

These heuristics minimize the number of range queries necessary during the run of DBSCAN in *cand*. If $\mathcal{C}^{cand} \neq \emptyset$, we add it to $\mathcal{C}_{k+1}$ and add *cand* to $\mathcal{S}_{k+1}$.

Step 2 is iteratively executed as long as the set of k-dimensional subspaces containing clusters is not empty.

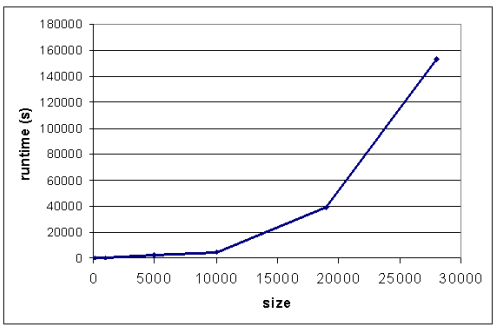
The most time consuming parts of our algorithm are all the partial range queries (range queries on arbitrary subspaces of the data space) necessary for the DBSCAN algorithm. As DBSCAN is applied to different subspaces, an index structure for the full-dimensional data space is not applicable. Therefore, we apply the approach of inverted files. Our algorithm provides an efficient index support for range queries on each single attribute in logarithmic time. For range queries on more than one attribute, we apply the range query to each separate attribute (index structure) and generate the intersection of all intermediate results to obtain the final result.

8.3.3 Experimental Evaluation

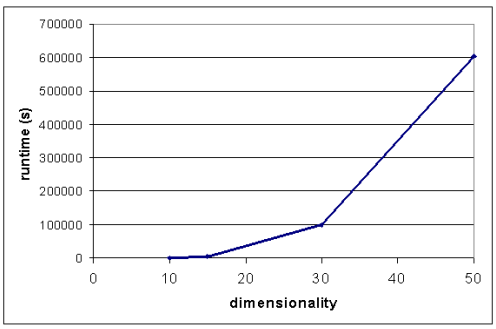
Efficiency

We evaluated the efficiency of SUBCLU using several synthetic data sets. All tests were run with $MinPts = 8$ and $\varepsilon = 2.0$.

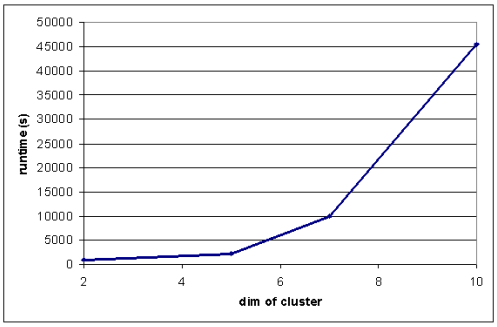
The scalability of SUBCLU against the size of the data set, the dimensionality of the data set and the dimensionality of the hidden subspace clusters are depicted in Figure 8.6. In all three cases, the runtime of SUBCLU grows with an at least quadratic factor. The reason for this scalability w.r.t. the size of the data set is that SUBCLU performs multiple range queries in arbitrary subspaces. As mentioned above, we can only support these queries using inverted files, since there is no index structure that can support partial range queries in average case logarithmic time. The scalability to the dimen-



(a) Scalability of SUBCLU against n.



(b) Scalability of SUBCLU against d.



(c) Scalability of SUBCLU against the maximum subspace cluster dimensionality.

Figure 8.6: Scalability of SUBCLU.

sionality of the data set and of the hidden subspaces can be explained by the *Apriori*-like bottom-up greedy algorithm underlying SUBCLU to navigate through the space of all possible subspaces.

Accuracy

To evaluate the effectivity of SUBCLU, we compared it with the subspace algorithm CLIQUE [AGGR98]. Since CLIQUE is a product of IBM and its code is not easy to obtain, we re-implemented CLIQUE according to [AGGR98]. In all accuracy experiments, we run CLIQUE with a broad range of parameter settings and took only the best results.

We applied SUBCLU and CLIQUE to several synthetic data sets which we

Table 8.1: Comparative evaluation of SUBCLU and CLIQUE: Summary of the results on synthetic data sets.

Name	d	dimension of subspace cluster	n	# of clusters	clusters found by SUBCLU	clusters found by CLIQUE
DS01	10	4	18999	1	1	1
DS02	10	4	27704	1	1	1
DS03	15	5,5,5	3802	3	3	1
DS04	15	3,5,7	4325	3	2	1
DS05	15	5,5,5	4057	3	3	1
DS06	15	4,4,6,7,7,10	2671	6	5	2

generated as described above. In each data set, several clusters are hidden in subspaces of varying dimensionality. The results are depicted in Table 8.1. In almost all cases, SUBCLU computed the artificial clusters whereas CLIQUE had difficulties in detecting all patterns properly. In addition, CLIQUE split usually connected clusters into several distinct clusters (not mentioned in the table).

We also applied SUBCLU to the Spellman data set (cf. Section 7.3) in order to find co-expressed genes.

SUBCLU found many interesting clusters in several subspaces of this data set. The most interesting clusters were found in the subspaces spanned by time slots 90, 110, 130, and 190 as well as time slots 190, 270, and 290. The functional relationships of the genes in the resulting clusters were investigated using the three biological criteria for functional relationships mentioned in Section 7.3.

The contents of four sample clusters in two different subspaces are depicted in Table 8.2. The first cluster (in subspace spanned by time slots 90, 110, 130, 190) contains several genes which are known to play a role during the cell cycle such as DOM34, CKA1, CPA1, and MIP6. In addition, the products of two genes in that cluster are part of a common protein complex. The second cluster contains the gene STE12, identified by [SSZ+98] as an important transcription factor for the regulation of the mitotic cell cycle. In addition, the genes CDC27 and EMP47 which have possible STE12-sites and are most likely co-expressed with STE12 are in that cluster. The third

Table 8.2: Contents of four sample clusters in different subspaces.

Gene Name	Function
Cluster 1 (subspace 90, 110, 130, 190)	
RPC40	subunit of RNA pol I and III, builds complex with CDC60
CDC60	tRNA synthetase, builds complex with RPC40
FRS1	tRNA synthetase
DOM34	protein synthesis, mitotic cell cycle
CKA1	mitotic cell cycle control
CPA1	control of translation
MIP6	RNA binding activity, mitotic cell cycle
Cluster 2 (subspace 90, 110, 130, 190)	
STE12	transcription factor (regulation of cell cycle)
CDC27	regulation of cell cycle, possible STE12-site
EMP47	Golgi membrane protein, possible STE12-site
XBP1	Transcription factor
Cluster 3 (subspace 90, 110, 130, 190)	
CDC25	starting control factor for mitosis
MYO3	control/regulation factor for mitosis
NUD1	control/regulation factor for mitosis
Cluster 4 (subspace 190, 270, 290)	
RPT6	protein catabolism; builds complex with RPN10
RPN10	protein catabolism; builds complex with RPT6
UBC1	protein catabolism; subunit of 26S protease
UBC4	protein catabolism; subunit of 26S protease
MRPL17	component of mitochondrial large ribosomal subunit
MRPL31	component of mitochondrial large ribosomal subunit
SNF7	direct interaction with VPS2
VPS4	mitochondrial protein; direct interaction with SNF7

cluster consists of the genes CDC25 (starting point for mitosis), MYO3 and NUD1 (known for an active role during mitosis) as well as various other transcription factors (e.g. CHA4, ELP3) required during the cell cycle. The fourth cluster contains several mitochondrion related genes which have similar functions. For example, the genes MRPL17, MRPL31, MRPL32, and MRPL33 (the last two are not listed in Table 8.2) are four mitochondrial large ribosomal subunits, the genes UBC1 and UBC4 are subunits of a certain protease, and the genes SNF7 and VPS4 are direct interaction partners. All gene products are located in mitochondria. This indicates a higher mito-

chondrial activity at these time spots which might be explained by a higher demand of biological energy during the cell cycle (the energy metabolism is located in mitochondria).

Let us note that the described four clusters are only a representative glance at the results SUBCLU yields when applied on the gene expression data set. Each cluster contains additional genes with yet unknown function. We also detected few clusters with no significant functional relationships among the grouped genes. However, most of the resulting clusters contained functional related genes, indicating that the detected co-expression is biological meaningful. Since most clusters also contain genes which have not yet any annotated function, the results of SUBCLU might propose a biologically interesting prediction for these genes. The overall results of SUBCLU on the Spellman data set are rather accurate, especially when the fact that this data set is fairly noisy is taken into account.

We also applied CLIQUE to the gene expression data set. We again tested a broad range of parameter settings and compared SUBCLU to the best results of CLIQUE. CLIQUE was not able to find any reasonable clusters in the gene expression data set possibly because it favors axis-parallel clusters. Thus, SUBCLU is much more suitable than CLIQUE due to the fact that the density-connected clustering notion underlying SUBCLU is able to detect arbitrarily shaped (subspace) clusters.

8.4 Density-Based Subspace Ranking

8.4.1 Motivation

SUBCLU, as introduced in Section 8.3, is rather effective in finding density-based subspace clusters. However, the most severe problem of SUBCLU and all other subspace clustering algorithms is the use of a global density threshold for the definition of clusters due to efficiency reasons. The use of a global density parameter yields the following handicaps:

1. The application of one global density threshold to subspaces of different dimensionality is rather unacceptable since the data space naturally increases exponentially with each dimension added to a subspace. Choosing a less strict density threshold results in getting a lot of low dimensional clusters or subspaces that are not really dense w.r.t. the dimensionality, while choosing a more strict density threshold has the contrary effect and cuts off all higher dimensional clusters. Thus, for subspace clustering, it would be highly desirable to adapt the density threshold to the dimensionality of the subspaces or even better to rely on a clustering notion that is independent from a globally fixed threshold. This problem is illustrated in Figure 8.7 (left): The objects of cluster C_1 in subspace $\{a_1\}$ are very densely packed, whereas they are rather sparsely located in subspace $\{a_1, a_2\}$. Even the density of the cluster C_2 is lower in subspace $\{a_1, a_2\}$ than in subspace $\{a_1\}$. If we want to find the 2D extension of C_2 as a cluster, the density threshold has to be specified rather low. As a consequence, the two clusters C_1 and C_2 in subspace $\{a_1\}$ may no longer be separable.

2. The application of one global density threshold to all clusters in one subspace of a fixed dimensionality is also rather unacceptable since the clusters may exceed different density parameters. A less strict threshold would detect one big cluster, whereas a more strict threshold would yield more clusters of smaller size. In addition, the information of nested clusters (clusters having sub-clusters of higher densities) is completely neglected. Therefore, for subspace clustering, a hierarchical approach would be desirable where the clustering notion does again not rely on a globally fixed threshold. This problem is illustrated in Figure 8.7 (right): A lower global density parameter would report the sets A, B, and C as clusters. The information of the nested clusters (C_1, C_2, and C_3) would be neglected. On the other hand, using a denser global threshold would detect clusters C_1, C_2, and C_3 but would neglect clusters A and B and the information that C is a nested cluster containing C_1, C_2, and C_3.

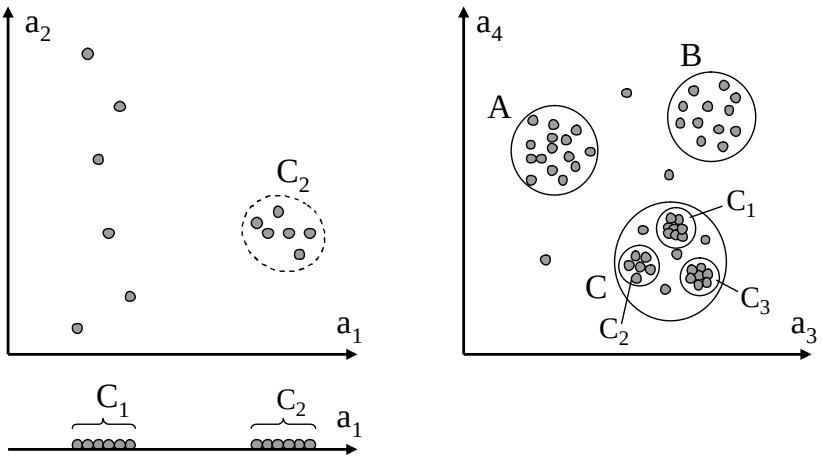


Figure 8.7: Problems with a global density parameter.

The first drawback is hard to tackle since the concepts of density-based subspace clustering are monotonic (cf. Section 8.2.2), and this is needed for efficiency reasons. However, at least the second drawback could be tackled if a hierarchical clustering algorithm, e.g. OPTICS (cf. Section 2.3), would be applied to the subspaces instead of a partitioning algorithm. In the rest of this section, we address the second problem of detecting clusters of different density in the same subspace.

8.4.2 General Idea

A first idea to get independent of a global parameter setting is to extend SUBCLU by hierarchical concepts, similar to the extension of the algorithm DBSCAN that resulted in the OPTICS algorithm. However, it is rather unclear how to navigate through the search space of all possible $O(2^d)$ subspaces. An *Apriori*-like greedy algorithm based on a monotonic concept used by CLIQUE or SUBCLU is preferable.

On the other hand, we could apply a hierarchical clustering algorithm such as OPTICS to some (say: "interesting") subspaces. All we need to do is to rate the interestingness of a subspace. The algorithm RIS (Ranking Interesting Subspaces for clustering), the details of which will be presented

in the following, works exactly like this. It is based on a quality criterion for subspaces that uses the density-based concepts of core points. Thus, RIS does not compute the subspace clusters hidden in the data directly, but can be seen as a preprocessing step for clustering, an advanced feature selection that selects several subspaces for clustering. The output of RIS is a list of subspaces sorted by descending quality values indicating how well the data points cluster in the according subspace. The clusters in the particular subspaces can be generated in a second step, using the clustering algorithm a user is most accomplished to.

8.4.3 Ranking Interesting Subspaces

Our approach to rate the interestingness of subspaces is based on the core point property. This property can be used for deciding about the interestingness of a subspace. Obviously, if a subspace contains no core point, it contains no dense region (cluster) and therefore contains no relevant information.

Observation 8.1 *The number of core points of a data set $\mathcal{D}$ (w.r.t. a given ε and MinPts) is proportional to the number of different clusters in $\mathcal{D}$ and/or the size of the clusters in $\mathcal{D}$ and/or the density of clusters in $\mathcal{D}$.*

This observation can be used to rate the interestingness of subspaces. However, simply counting all the core points for each subspace delivers not enough information. Even if two subspaces contain the same number of core points, the quality may differ a lot. Dense regions also contain border points, i.e. points which are not core points themselves but lie within the ε-neighborhood of a core point and are thus a vital part of the dense region. Therefore, it is not only interesting how many core points a subspace contains but also how many objects lie within the ε-neighborhood of these core points.

Definition 8.9 (count-value of a subspace)
The count-value *(w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$) of a subspace $S \subseteq \mathcal{A}$, denoted by count$[S]$, is the sum of all points lying in the ε-neighborhood of*

all core points (w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$) in the subspace S, formally:

$$count[S] = \sum_{p \in \mathcal{D},\, \textsc{Core}^S_{den}(p)} |\mathcal{N}^S_\varepsilon(p)|.$$

If we measure the interestingness of a subspace according to its $count[S]$ value and rank all subspaces according to this quality value, a severe problem is not addressed.

Recall from Observation 7.2 in Section 7.1 that the ε-neighborhoods of the core points tend to exceed the boundaries of the data space with increasing dimensionality. As a consequence, naturally with each dimension, the number of expected points in the ε-neighborhood of a core point decreases. Thus, this naive quality value favors lower dimensional subspaces over higher dimensional ones. A first solution to overcome this problem is that we introduce a scaling co-efficient that takes the dimensionality of the subspace into account. We take the ratio between the $count[S]$ value and the "virtual" count value of S we would get if all data objects were uniformly distributed in S.

For that purpose, we compute the volume of a d-dimensional ε-neighborhood, denoted by Vol^d_ε. If $dist$ is the L_∞-norm, Vol^d_ε is a hypercube and can be computed by $Vol^d_\varepsilon = (2\varepsilon)^d$ or if $dist$ is the Euclidian distance (L_2-norm), Vol^d_ε is a hyper-sphere and can be computed as given below:

$$Vol^d_\varepsilon = \frac{\sqrt{\pi^d}}{\Gamma(d/2+1)} \cdot \varepsilon^d$$

where $\Gamma(x+1) = x \cdot \Gamma(x), \quad \Gamma(1) = 1 \quad$ and $\Gamma(\frac{1}{2}) = \sqrt{\pi}$.

If we further assume that the points are normalized within $[0, MAX]^d$, i.e. MAX is the maximum value of each attribute, and are uniformly distributed, we expect n objects in the volume MAX^d. The number of points expected to be in Vol^d_ε is

$$\frac{Vol^d_\varepsilon \cdot n}{MAX^d}.$$

Since the number of range queries in a particular subspace is n, we scale the $count$-value by n times the number of points expected to be in Vol^d_ε.

Thus, the quality of a subspace can be computed as given in the following definition.

Definition 8.10 (subspace quality)
The quality of a subspace $S \subseteq \mathcal{A}$, measuring the interestingness of S is defined by:

$$\textsc{Quality}(S) = \frac{count[S]}{n \cdot \frac{Vol_\varepsilon^{dim[S]} \cdot n}{MAX^{dim[S]}}}$$

This quality value would still favor lower dimensional subspaces. Due to the above mentioned phenomenon, the ε-neighborhoods of many points most likely exceed the boundaries of the data space when the dimensionality increases. As a consequence, the estimation of the volume of these ε-neighborhoods using $Vol_\varepsilon^{dim[S]}$ is inadequate in higher dimensional spaces.

In [BBKK97] the authors show that the average volume of the intersection of the data space and a hyper-sphere with radius ε can be expressed as the integral of a piecewise defined function, integrated over all possible positions of the ε-neighborhood, i.e the core points. For our implementation, we choose a less complex, commonly used heuristics to eliminate this effect based on periodical extensions of the data space (cf. Section 8.4.4 for details). Using these heuristics, the quality criterion is robust against the dimensionality of the subspace.

For two subspaces $U, V \subseteq \mathcal{A}$ with $U \supset V$ this quality criterion has two complementary effects which are summarized in the following lemmata:

Lemma 8.6
Let $U \supset V$. Then the following inequality holds:

$$count[U] \leq count[V].$$

Proof. $\forall p, x \in \mathcal{D}$:

$$\textsc{Core}_{den}^{U}(p) \wedge x \in \mathcal{N}_\varepsilon^U(p)$$

$$\begin{array}{cl} \Leftrightarrow & \textsc{Core}^U_{den}(p) \wedge dist(\pi_U(p), \pi_U(x)) \leq \varepsilon \\ \overset{U \supset V,\ \text{Lemma 8.1}}{\Longrightarrow} & \textsc{Core}^V_{den}(p) \wedge dist(\pi_V(p), \pi_V(x)) \leq \varepsilon \\ \Leftrightarrow & \textsc{Core}^V_{den}(p) \wedge x \in \mathcal{N}^V_\varepsilon(p). \end{array}$$

Thus, each object x contributing to $count[U]$ also contributes to $count[V]$. On the other hand, the reverse implication does obviously not hold in general. In summary, we have $count[U] \leq count[V]$. □

Lemma 8.7
Let $U \supset V$.

$$count[U] = count[V] \Rightarrow \textsc{Quality}(U) \geq \textsc{Quality}(V).$$

Proof. *Through simple algebraic transformations we get*

$$\textsc{Quality}(S) = \frac{count[S] \cdot MAX^{dim[S]}}{n^2 \cdot Vol_\varepsilon^{dim[S]}}.$$

Since $U \supset V$, and we can assume $MAX \geq 2\varepsilon$, it follows that $MAX^{dim[S]}$ grows faster with increasing dimensionality than $Vol_\varepsilon^{dim[S]}$. Thus, we can conclude from the assumption ($count[U] = count[V]$) that $\textsc{Quality}(U) \geq \textsc{Quality}(V)$. □

The lemmata state that while navigating through the subspaces bottom-up, the *count* value decreases (cf. Lemma 8.6) until at a certain point the core points loose their core point property due to the addition of irrelevant features. The consequence of adding irrelevant features is that the quality decreases. On the other hand, as long as this is not the case, i.e. the *count* values are stable, the features are relevant for the clusters and the quality increases (cf. Lemma 8.7). Obviously, this is a desirable behavior of the quality measure.

8.4.4 Algorithm RIS

Given a set of objects $\mathcal{D}$ and density parameters ε and *MinPts*, RIS finds all interesting subspaces and presents them to the user ordered by relevance. For

```
algorithm RIS(SetOfObjects 𝒟, Real ε, Integer MinPts)
    Subspaces := ∅ ;
    for i from 1 to n do
        Object := SampleObjects.get(i);
        RelevantSubspaces := GenerateSubspaces(Object,SetOfObjects);
        Subspaces.add(RelevantSubspaces);
    end for
    Subspaces.prune();
    Subspaces.sort();
```

Figure 8.8: The RIS algorithm.

each object, RIS computes a set of relevant subspaces. All these sets are then merged. A pruning and sorting procedure is applied to the resulting set of subspaces. The pseudo code of the algorithm RIS is given in Figure 8.8. For each object $o \in \mathcal{D}$, all subspaces in which the core point condition holds for o are computed. This step will be described below in more detail. Let us note that the algorithm can also be applied to a sample of $\mathcal{D}$, e.g. for performance reasons (see the experimental evaluation in Section 8.4.5). For each detected subspace, statistical data is accumulated. The detected subspaces are pruned according to certain criteria. In Section 8.4.4, these criteria will be discussed. Finally, the subspaces are sorted for a more comprehensible user presentation. Then the clustering in these subspaces can be generated by any clustering algorithm.

Efficient Generation of Subspaces

For a given object $o \in \mathcal{D}$, the method GenerateSubspaces finds all subspaces S in which the core point condition holds for o w.r.t. ε and $MinPts$. Formally, it computes the following set:

$$K_o := \{T \subseteq \mathcal{A} \mid \quad |\mathcal{N}_\varepsilon^T(o)| \geq MinPts\}.$$

The problem of finding the set K_o is equivalent to the problem of determining all frequent itemsets in the context of mining association rules [AS94] when using the L_∞-norm as distance function and thus can be computed rather efficiently. Let us note that the use of L_∞-norm is no serious

constraint. The only difference is that by using the L_∞-norm we may find additional core points because we will find few additional points in the ε-neighborhood of some points and thus additional subspaces. However, these additional subspaces get low quality values anyway.

For each $x \in \mathcal{D}$ a transaction $T_x \subseteq \mathcal{A}$ is defined such that

$$a_i \in T_x \Leftrightarrow |\, \pi_{\{a_i\}}(x) - \pi_{\{a_i\}}(o) \,| \leq \varepsilon \quad \text{for all } i \in \{1, \ldots, d\}.$$

Lemma 8.8

Let K_o be defined as given above. For all $o \in \mathcal{D}$, the following holds.

$$K_o = \{T \subseteq \mathcal{A} \mid \text{Supp}_{\mathcal{D}}(T) \geq \frac{MinPts}{n}\}$$

where

$$\text{Supp}_{\mathcal{D}}(T) = \frac{|\{x \in \mathcal{D} \,|\, T \subseteq T_x\}|}{n}.$$

Proof.

$$\begin{aligned}
&T \subseteq \mathcal{A} \wedge |\mathcal{N}_\varepsilon^T(o)| \geq MinPts \\
&\quad \Leftrightarrow T \subseteq \mathcal{A} \wedge |\{x \in \mathcal{D} \mid dist_{L_\infty}(\pi_T(o), \pi_T(x)) \leq \varepsilon\}| \geq MinPts \\
&\quad \overset{dist=L_\infty}{\Leftrightarrow} T \subseteq \mathcal{A} \ \wedge \\
&\qquad |\{x \in \mathcal{D} \,|\, \forall i \in \{1, \ldots, d\} : a_i \in T \Rightarrow |\pi_{\{a_i\}}(o) - \pi_{\{a_i\}}(x)| \leq \varepsilon\}| \\
&\qquad\quad \geq MinPts \\
&\quad \Leftrightarrow T \subseteq \mathcal{A} \wedge |\{x \in \mathcal{D} \,|\, T \subseteq T_x\}| \\
&\qquad \geq MinPts \\
&\quad \Leftrightarrow T \subseteq \mathcal{A} \wedge \text{Supp}_{\mathcal{D}}(T) \geq \frac{MinPts}{n}.
\end{aligned}$$

□

The method GenerateSubspaces extends the familiar *Apriori* algorithm [AS94] in accumulating the statistical information for measuring the subspace quality, using the monotonicity of the core point condition (cf. Lemma 8.1). As mentioned before, we are extending the data space periodically to address the problems stated in Observations 7.2, ensuring that all ε-neighborhoods

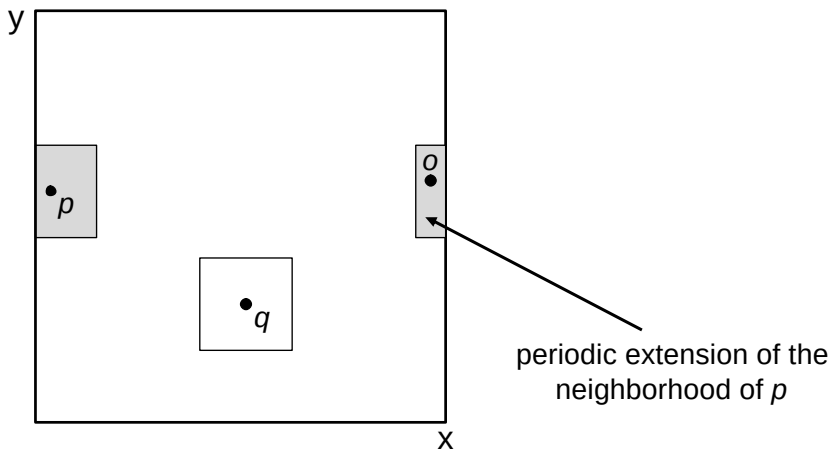


Figure 8.9: Illustration of the periodic extension of the data space ($dist = L_\infty$).

have the same size. This can be done very easily by changing the way the transactions are defined. Instead of only checking if $|\pi_{a_i}(x) - \pi_{a_i}(o)| \leq \varepsilon$, we have to check if $|\pi_{a_i}(x) - \pi_{a_i}(o)| \leq \varepsilon$ or $|\pi_{a_i}(x) - \pi_{a_i}(o)| \geq attrRange - \varepsilon$. A 2-dimensional example is illustrated in Figure 8.9. The ε-neighborhood (in case of $dist = L_\infty$ the geometric interpretation is a hyper-cube) of point p exceeds the data space along the y-axis and is periodically extended as depicted, and thus, $o \in \mathcal{N}_\varepsilon^{\{x,y\}}(p)$. The ε-neighborhood of q obviously needs no extension because it does not exceed the boundaries of the data space.

Pruning of Redundant Subspaces

As we are only interested in the subspaces which provide the most information, we can perform the following two downward pruning steps to eliminate redundant subspaces:

First, if there exists a $(k+1)$-dimensional subspace S with higher quality than the k-dimensional subspace T and $S \supset T$, we delete T because T is redundant (cf. Lemma 8.7).

For the second pruning step, we assume that for a given data set the k-dimensional subspace S reflects the clustering in that special data set in

a best possible way. Thus, its quality value and the quality values of all its $(k-1)$-dimensional subspaces $T_1, \ldots, T_m$ is high. On the other hand, if we combine one of these $(k-1)$-dimensional subspaces $T_1, \ldots, T_m$ with another 1-dimensional subspace with lower quality, the quality of the resulting k-dimensional subspace can still be good. But as we know that it does not reflect the clustering in the best possible way, we are not interested in this k-dimensional subspace. The following heuristic pruning eliminates such subspaces. Let S be a k-dimensional attribute space and $S_{k-1} := \{T \mid T \subset S \wedge \dim[T] = k-1\}$ be the set of all $(k-1)$-dimensional subspaces of S. Let $\overline{count}$ be the mean count value of all $T \in S_{k-1}$ and $\overline{s}$ be the standard deviation. Let $maxdiff := \max_{T \in S_{k-1}} (\,|\,count[T] - \overline{count}|\,)$ be the maximum deviation of the count-values of all $T \in S_{k-1}$ from the mean count-value. Then, the so-called *bias*-value can be computed as follows: $bias = \frac{\overline{s}}{maxdiff}$. If this bias-value falls below a certain threshold, we prune the k-dimensional subspace S. Experimental evaluations indicate that 0.56 is a good value for this bias-criterion.

Determination of Density Parameters

A heuristic method, which is experimentally shown to be sufficient, suggests $MinPts \approx \ln(n)$ where n is the size of the database. Then, ε must be picked, depending on the value of *MinPts*. In [EKSX96] a simple heuristics is presented to determine the ε of the "thinnest" cluster in the database (for a given *MinPts*). But as we do not know beforehand in which subspaces clusters will be found, we cannot determine ε to find a single subspace with one particular clustering. Quite the contrary, we want to choose the parameters such that RIS detects subspaces which might have clusters of different density and different dimensionality.

However, we can determine an upper bound for ε for a given value of *MinPts*. If we take uniform distribution as worst case, the ε-neighborhood of an object should not contain more than $(MinPts - 1)$ objects in the full-dimensional space. Otherwise, all objects are core points. In case of the L_∞-norm, an upper bound for ε can be computed as follows:

$$n \cdot \frac{Vol_\varepsilon^d}{attrRange^{dim}} < MinPts \quad \overset{L_\infty}{\Longrightarrow} \quad \varepsilon < \frac{attrRange}{2} \cdot \sqrt[dim]{\frac{MinPts}{n}}$$

where $dim = d$. If we have any knowledge about the dimensionality of the subspaces we want to find, we can further decrease the upper bound by setting dim to the highest dimension of such a subspace.

This upper bound is very rough. Nevertheless, it provides a good indication for the choice of ε. Indeed, it empirically turned out that $upperbound/4$ is a reasonable choice for ε. Experiments on synthetic data sets show that our suggested criteria for the choice of the density parameters are sufficient to detect the relevant subspaces containing clusters.

8.4.5 Experimental Evaluation

We tested RIS using several synthetic as well as the Spellman gene expression data set described in 7.3. The experiments were run on a workstation with a 1.7 GHz CPU and 2 GB RAM.

The synthetic data sets were generated by a self-implemented data generator. It permits to control the size and structure of the generated data sets through parameters such as number and dimensionality of subspace clusters, dimensionality of the feature space and density parameters for the whole data set as well as for each cluster. In a subspace that contains a cluster, the average density of data points in that cluster is much larger than the density of points not belonging to the cluster in this subspace. In addition, it is ensured that none of the synthetically generated data sets can be clustered in full-dimensional space.

A subsequent clustering of the data sets in the detected subspaces was performed for each experiment using the above mentioned algorithm OPTICS to validate the interestingness of the subspaces computed by RIS.

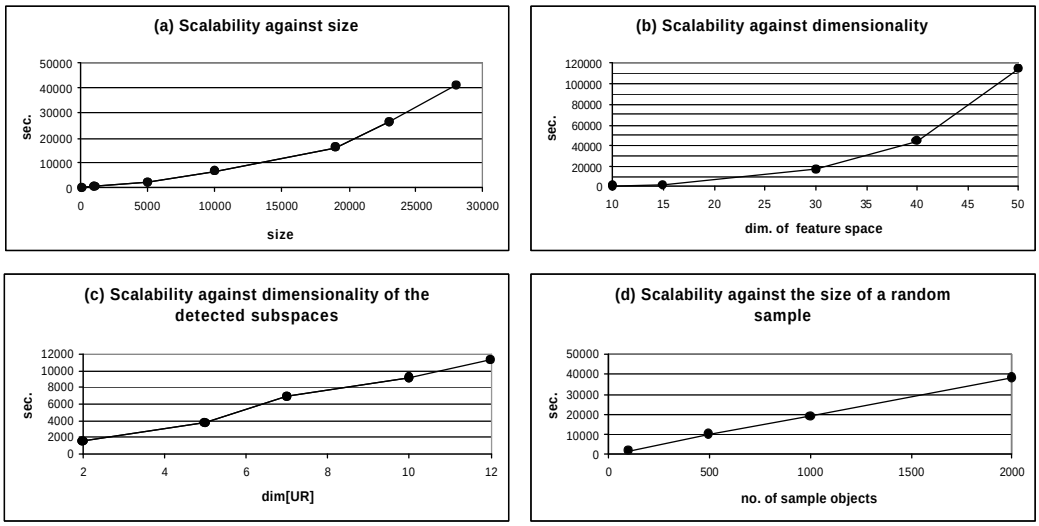


Figure 8.10: Efficiency evaluation.

Efficiency

The results of the efficiency evaluation are depicted in Figure 8.10. This evaluation is based on several synthetic data sets. The experiments were run with $MinPts = \ln(n)$ and ε chosen, as suggested in Section 8.4.4. All run times are in seconds.

RIS scales well to the dimensionality of the relevant subspaces. With increasing dimensionality of the relevant subspaces, the runtime of RIS grows with a linear factor. On the other hand, the scalability of RIS to the size n and the dimensionality d of the input data set is not linear. With increasing n and d, the runtime of RIS grows with an at least quadratic factor for rather large n and d, respectively. The reason for this scalability vs. the size n is that RIS performs multiple range queries without any index support, due to the fact that the ε-neighborhoods of all points in arbitrary subspaces have to be computed. However, there is no index structure to efficiently support range queries in arbitrary subspaces. The observed scalability with respect to d can be explained by the *Apriori*-like navigation through the search space of all subspaces.

Speed-up for Large Data Sets

Since the runtime of RIS is rather high especially for large data sets, we applied random sampling to accelerate our algorithm. Figure 8.10 shows that for a large data set of $n = 750,000$ data objects, sampling yields a rather good speed-up. The data set contained two overlapping four-dimensional subspace clusters, containing approximately 400,000 and 350,000 points. Even using only 100 sample points, RIS had no problem to detect the subspaces of these two clusters. For all sample sizes, these subspaces had by far the highest quality values. Further experiments empirically show that random sampling can be successfully applied to RIS in order to speed-up the runtime of this algorithm, paying a minimum loss of quality.

Accuracy

Synthetic Data Sets. We evaluated the effectiveness of RIS using several synthetic data sets of varying dimensionality. The data sets contained between two and five overlapping clusters in varying subspaces. In all experiments, RIS detected the correct subspaces in which clusters exist and assigned the highest quality values to them. All higher dimensional subspaces which were generated, were removed by the pruning procedures.

Gene Expression Data. We also applied RIS to the Spellman data set. The two top-ranked subspaces were the subspace spanned by the time spots 90, 110, 130, and 190 and the subspace spanned by the time spots 190, 270 and 290. Both subspaces played also a central role in the evaluation of the algorithm SUBCLU (cf. Section 8.3.3). A clustering using OPTICS in these two top-ranked subspaces provided several clusters and in fact more information than SUBCLU yields. This is due to the use of a hierarchical clustering algorithm in the detected subspaces. For example, the genes MRPL17, MRPL31, MRPL32, and MRPL33 (four mitochondrial large ribosomal subunits), were clustered together with other mitochondrial proteins SNF7 and VPS4 (which are direct interaction partners) by SUBCLU. However, several

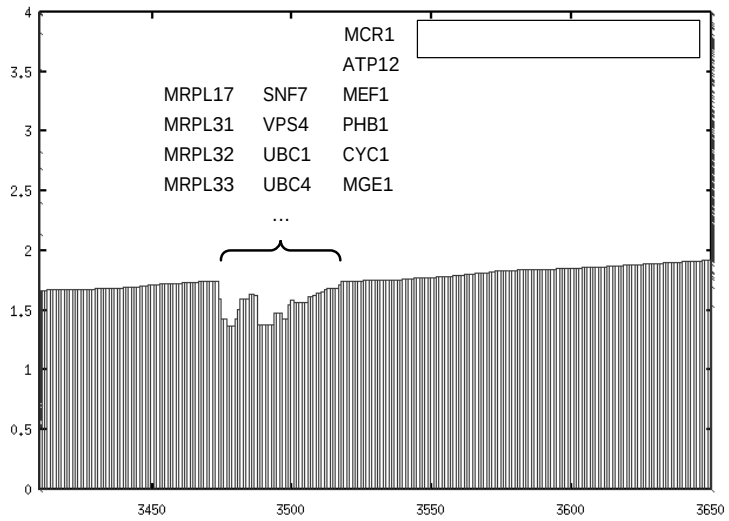


Figure 8.11: Part of the reachability plot generated by OPTICS in the subspace, ranked second by RIS.

other genes that code for mitochondrial proteins (e.g. MEF1, PHB1, CYC1, MGE1, ATP12) could be added to this cluster because of the information OPTICS yielded in this subspace. Figure 8.11 illustrates the part of the cluster ordering generated by OPTICS in the particular subspace. It can be seen that the additional genes, obviously still a virtual part of the cluster, are less dense than the core part of the cluster. To detect the entire nested cluster, the global parameter setting for the SUBCLU run in Section 8.3.3 was too strict, i.e. the ε-value to was too small. However, running SUBCLU with a higher ε-value blurs the clusters found in Section 8.3.3 by non-related genes, i.e. noise points.

A second example of the information gain is the cluster of which an excerpt is depicted in Table 8.3. The cluster was found in the same subspace, spanned by the time spots 90, 110, 130, and 190 and contains several transcription related genes that directly interact with each other. It was not detected by SUBCLU because it does not fit the density threshold used for the SUBCLU run. However, it is a significant valley in the reachability plot generated by OPTICS in that subspace and thus a true cluster. The functional relationships of the contained genes is biologically meaningful and

Table 8.3: A cluster missed by SUBCLU, but detected by RIS/OPTICS.

Gene Name	Function
RRP3	RNA splicing, builds complex with NPL3
NPL3	RNA splicing, builds complex with RRP3
TFA1	transcription elongation factor
SPT5	part of transcription elongation factor complex (TEFC)
CDC73	part of TEFC, builds complex with CKB1
CKB1	cell cycle transition gene, builds complex with CDC73

important.

In summary, RIS detects several subspaces containing several biologically relevant co-expressions. All significant clusters SUBCLU has found were reproduced by the combined application of RIS and OPTICS. Furthermore, the application of the hierarchical algorithm OPTICS yielded additional insights such as extended nested clusters and more clusters showing different densities. By outperforming SUBCLU, the combined application of RIS and OPTICS also yields superior accuracy than CLIQUE.

8.5 Summary and Discussion

In this chapter, we proposed two novel algorithms for subspace clustering based on the density-based clustering notion.

The algorithm SUBCLU (density-based SUBspace CLUstering) is an efficient extension of the density-based clustering algorithm DBSCAN for the subspace clustering problem. It detects all subspace clusters that would have been found if DBSCAN would have been applied exclusively to each subspace. Based on the monotonicity of density connected sets, SUBCLU excludes all subspaces that cannot contain any density connected cluster from further computation and is thus much more efficient than an exhaustive search, i.e. the application of DBSCAN to all possible subspaces. An experimental evaluation of SUBCLU, using gene expression data, empirically shows the superior performance over existing subspace clustering algorithms, such as CLIQUE, in terms of accuracy and information gain.

Since the application of a global density threshold to the clusters of one single subspace is rather restrictive, and a hierarchical subspace clustering is intended in several applications, we proposed RIS (Ranking Interesting Subspaces for clustering). RIS does not directly compute subspace clusters but ranks the subspaces according to their interestingness in terms of clustering quality. It relies on a quality criterion for subspaces and generates a sorted list of interesting subspaces. Any clustering algorithm can be applied to the resulting subspaces of interest. We evaluated RIS in combination with OPTICS (to compute a hierarchical clustering in some resulting subspaces), applied on the gene expression data set, and empirically showed that RIS can further outperform SUBCLU in terms of the information gained.

Chapter 9

Correlation Clustering

Beside point density, a second kind of hidden information that may be interesting to users are local correlations in a data set. In this chapter, we propose a density-based algorithm to the correlation clustering problem called *4C*. First, we discuss recent approaches related to correlation clustering in Section 9.1. Thereafter, we present the foundations of density-based correlation clustering underlying 4C in Section 9.2 which rely on a combination of the density-based clustering concepts and a suitable primitive to measure the correlation. In Section 9.3, the details of the 4C algorithm are described. A broad experimental evaluation of 4C is presented in Section 9.4. Two modifications of the concepts of 4C that produce a density-based solution for pattern-based clustering and for projected clustering are proposed in Section 9.5. The concepts described in this chapter are major extensions of the material published in [BKKZ04]. Section 9.6 concludes the chapter with a short summary.

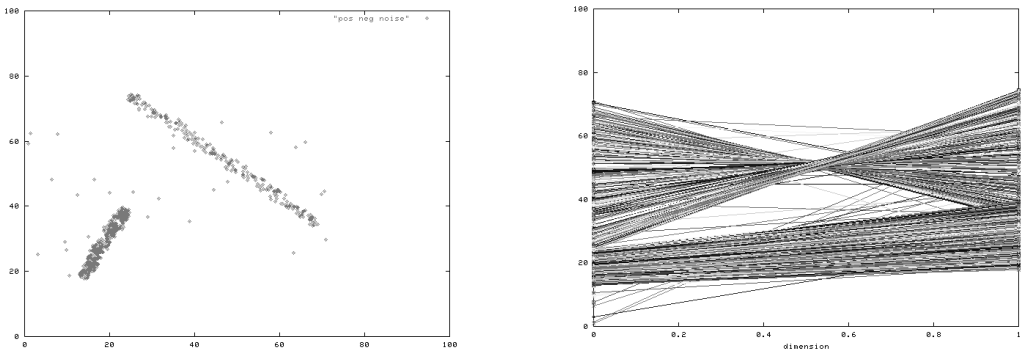
9.1 Motivation and Related Work

Beside point density, a 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 (PCA), also known as Karhunen Loèwe transformation. 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/her disease or the co-regulation of gene expression) or to gain financial advantage (e.g. in stock quota analysis), etc.

However, traditional methods such as PCA are restricted, because they are global and can only be applied to the entire data. 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 entire data set can be weak, even if for local subsets of the data strong correlations exist. Figure 9.1 shows a simple example, where two subsets of 2-dimensional points exhibit different correlations. We use the transposed view (cf. Section 7.2) to visualize simple correlations.

Projected clustering algorithms such as PROCLUS [APW+99] and DOC [PJAM02] are restricted to find axis-parallel dense projections. However, correlations may be arbitrarily oriented, i.e. the dense projections are not axis-parallel.

To the best of our knowledge, both concepts of clustering (i.e. finding



(a) 2D view. **(b)** Transposed view.

Figure 9.1: 1-dimensional correlation lines.

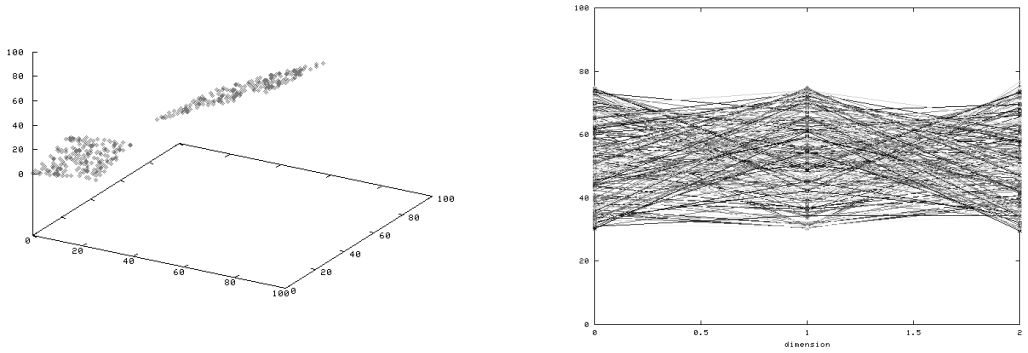
densely populated subsets of the data) and correlation analysis have not yet been addressed as a combined task for data mining. The most relevant related approach is ORCLUS [AY00]. It is a k-means like approach that iteratively optimizes the clustering quality. Each cluster C_i is represented by its centroid and is associated with a set of pairwise orthonormal vectors S_i that span an arbitrarily oriented subspace of the cluster. In each iteration, the points of the database are first assigned to the nearest medoid. The computation of the distance between a point and the medoid of cluster C_i is adopted according to S_i, i.e. is the distance between the two points in subspace spanned by S_i rather than in full-dimensional space. After that, new spanning vectors S_i are computed using PCA (only the l eigenvectors with the smallest eigenvalues are taken; l is an input parameter). The medoid of the new cluster C_i is computed according to S_i.

The problems of ORCLUS are the typical drawbacks of optimization-based clustering methods. First, the user has to specify the number of clusters k in advance. If this guess does not correspond to the actual number of clusters the results of ORCLUS deteriorate. For example, if k is too small, the locality of the analyzed correlations is usually too coarse, i.e. the number of points taken into account for correlation analysis is too large. A second problem is noisy data. In this case, the clusters found by ORCLUS are far from optimal since ORCLUS assigns each point to a cluster and thus cannot handle noise efficiently. An additional problem is that all clusters

must have the user-specified intrinsic dimensionality l (for each cluster C_i, l eigenvectors are added to S_i). In real data sets, this is a rather coarse generalization, because in fact, the intrinsic dimensionality of the clusters (i.e. correlations) may differ significantly. As a consequence, the assignment of points to clusters may be even more ambiguous.

As mentioned previously, pattern-based algorithms can find some specialized correlations in a data set. The pioneering approach for pattern-based clustering is presented in [YWWY02], introducing the so-called δ-clusters model and the algorithm FLOC to compute near-optimal δ-clusters. The transposed view of the data is used to show the correlations which are captured by this δ-cluster model. A cluster is regarded as a subset of objects and attributes for which the participating objects show the same (or a similar) tendency (pattern) rather than being close to each other on the associated attributes. The δ-cluster model concentrates on two forms of coherence, namely shifting (or addition) and amplification (or production). In the case of amplification coherence for example, the vectors representing the objects must be multiples of each other. The authors state that this can easily be transformed to the problem of finding shifting coherent δ-cluster by applying logarithmic function to each object. Thus, they focus on finding shifting coherent δ-clusters and introduce the metric of residue to measure the coherency among objects of a given cluster. An advantage is that thereby they can easily handle missing attribute values. However, the δ-cluster model limits itself to a very special form of correlation where all attributes are positively linear correlated. It does not include negative correlations or correlations where one attribute is determined by two or more other attributes. In these cases, searching for a trend is no longer possible as can be seen in Figure 9.2(b). As noted previously, such complex dependencies cannot be illustrated by transposed views of the data. The same considerations apply for the very similar p-cluster model introduced in [WWYY02] and two extensions presented in [PZC+03] and [LW03].

In the following, we develop 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) 3D view. **(b)** Transposed view of one plane.

Figure 9.2: 2-dimensional correlation planes.

a *correlation connected cluster*. Its correlation will be hidden locally in the data set and cannot be detected by global techniques. Figures 9.1 and 9.2 show simple examples how correlation connected clusters can look like. In Figure 9.1 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 9.2 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 noted above, the transposed view is not capable of appropriately visualizing such a complex linear dependency (cf. Figure 9.2(b)). Obviously, there is no common pattern visible.

9.2 Foundations of Connected Correlation Clustering

9.2.1 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 in-

trinsic 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 $\mathbf{M} = [m_{ij}]$ with $m_{ij} = \sum_{q \in S} \pi_{\{a_i\}}(q) \cdot \pi_{\{a_j\}}(q)$ of a considered point set S, and decomposes it into an orthonormal matrix $\mathbf{V}$ called eigenvector matrix and a diagonal matrix $\mathbf{E}$ called eigenvalue matrix such that $\mathbf{M} = \mathbf{VEV^T}$. The eigenvectors represent the principal axes of the data set, 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 9.1 (λ-dimensional linear correlation set)
Let $S \subseteq \mathcal{D}$, $\lambda \in \mathbb{N}$ ($\lambda \leq d$), $EV = e_1, ..., e_d$, the eigenvalues of S in descending order (i.e. $e_i \geq e_{i+1}$) and $\delta \in \mathbb{R}$ ($\delta \approx 0$). S forms a λ-dimensional linear correlation set *w.r.t. δ if at least $(d - \lambda)$ eigenvalues of S are close to zero, formally:*

$$\textsc{CorSet}_\delta^\lambda(S) \Leftrightarrow |\{e_i \in EV \mid \Omega(e_i) \leq \delta\}| \geq d - \lambda,$$

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

This condition states that the variance of S along $(d - \lambda)$ principal axes is low and therefore the points of S form a λ-dimensional hyperplane. If we drop the index λ and speak of a correlation set in the following, we mean a λ-dimensional linear correlation set where λ is not specified but fix.

Definition 9.2 (Correlation dimension)
Let $S \in \mathcal{D}$ be a linear correlation set w.r.t. $\delta \in \mathbb{N}$. The number of eigenvalues with $\Omega(e_i) > \delta$ is called correlation dimension*, denoted by* $\textsc{CorDim}(S)$.

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

9.2.2 Clusters as Correlation Connected Sets

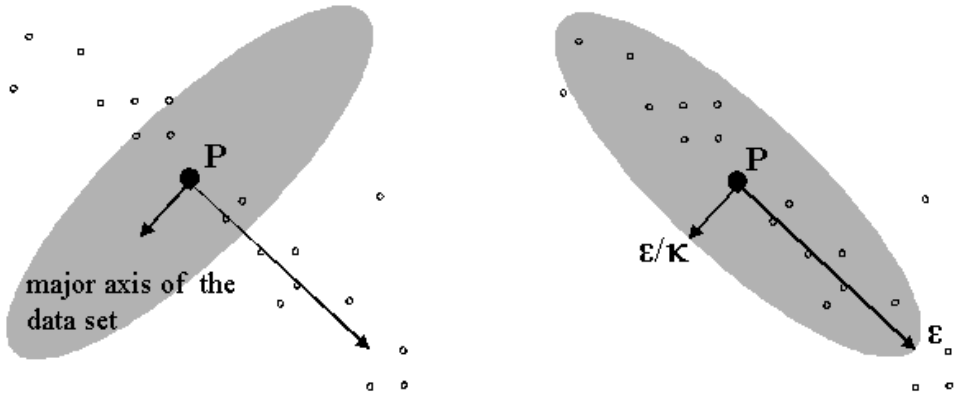
A correlation connected cluster can be regarded as a maximal set of density-connected points that exhibit uniform correlation. We can formalize the concept of correlation connected sets by merging the concepts of density connected sets (cf. Definition 2.7) and correlation sets (cf. Definition 9.1). The intuition of our formalization is to consider those points as core points of a cluster which have an appropriate correlation dimension in their neighborhood. Therefore, we associate each point p with a similarity matrix $\mathbf{M}_p$ which is determined by PCA of the points in the ε-neighborhood of p. For convenience, we call $\mathbf{V}_p$ and $\mathbf{E}_p$ 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 $\hat{\mathbf{M}}_p$ for the efficient search of such points.

We start with the formal definition of the covariance matrix $\mathbf{M}_p$ associated with a point p.

Definition 9.3 (covariance matrix)
Let $p \in \mathcal{D}$. The matrix $\mathbf{M}_p = [m_{ij}]$ with

$$m_{ij} = \sum_{q \in \mathcal{N}_\varepsilon(p)} \pi_{\{a_i\}}(q) \cdot \pi_{\{a_j\}}(q) \qquad (1 \leq i, j \leq d)$$

is called the covariance matrix *of the point p. $\mathbf{V}_p$ and $\mathbf{E}_p$ (with $\mathbf{M}_p$ =*



(a) (b)

Figure 9.3: Correlation ε-neighborhood of a point p according to (a) $\mathbf{M}_p$ and (b) $\hat{\mathbf{M}}_p$.

$\mathbf{V}_p\mathbf{E}_p\mathbf{V}_p^{\mathbf{T}}$*), as determined by PCA of* $\mathcal{N}_\varepsilon(p)$*, are called the eigenvectors and eigenvalues of the point* p*, respectively.*

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

$$dist_{\mathbf{M}_p}(p,q) = \sqrt{(p-q)\mathbf{M}_p(p-q)^{\mathbf{T}}} \quad \text{where } p,q \in \mathcal{D}.$$

Figure 9.3(a) shows the set of points which lies in an ε-neighborhood of the point, using $\mathbf{M}_p$ 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 to 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 9.3(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, for example, 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 ε. 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 9.4 (correlation similarity matrix of a point)
Let $p \in \mathcal{D}$ and $\mathbf{V}_p$, $\mathbf{E}_p$ 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 $\hat{\mathbf{E}}_p$ with entries $\hat{e}_i$ $(i = 1, \ldots d)$ is computed from the eigenvalues $e_1, \ldots, e_d$ in $\mathbf{E}_p$ according to the following rule:

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

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

$$dist_p(p, q) = \sqrt{(p - q) \cdot \hat{\mathbf{M}}_p \cdot (p - q)^{\mathbf{T}}}.$$

Figure 9.3(b) shows the ε-neighborhood according to the correlation similarity matrix $\hat{\mathbf{M}}_p$. 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 9.3.1 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 points and correlation reachability. However, in order to define correlation connectivity as a symmetric relation, we face the problem that the similarity

(a) (b)

Figure 9.4: Symmetry of the correlation ε-neighborhood: (a) $p \in \mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_q}(q)$. (b) $p \notin \mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_q}(q)$.

measure in Definition 9.4 is not symmetric, because $dist_p(p,q) = dist_q(q,p)$ does in general not hold (cf. Figure 9.4(b)). Symmetry, however, is important to avoid ambiguity of the clustering result. If an asymmetric 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 point) because the symmetry of density connectivity depends on the symmetry of direct density reachability for core points. 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.

Definition 9.5 (general correlation distance)
The general correlation distance *between two points $p, q \in \mathcal{D}$, denoted by $dist_{corr}$, is defined as the maximum of the correlation similarity measure between p and q according to p and according to q, formally:*

$$dist_{corr}(p,q) = \max\{dist_p(p,q), dist_q(q,p)\}.$$

Lemma 9.1 *The general correlation distance as defined in Definition 9.5 is symmetric.*

Proof. *Obvious from Definition 9.5.* □

Based on this new symmetric similarity measure $dist_{corr}$, we define the correlation ε-neighborhood as a symmetric concept.

Definition 9.6 (correlation ε-neighborhood)
Let $\varepsilon \in \mathbb{R}$. The correlation ε-neighborhood *of a point $o \in \mathcal{D}$, denoted by $\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_o}(o)$, is defined by:*

$$\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_o}(o) = \{x \in \mathcal{D} \mid dist_{corr}(o,x)\} \leq \varepsilon\}.$$

The symmetry of the correlation ε-neighborhood is illustrated in Figure 9.4. A point p is only contained in $\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_q}(q)$ if q is also contained in $\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_p}(p)$. Correlation core points can now be defined as follows.

Definition 9.7 (correlation core point)
Let $\varepsilon, \delta \in \mathbb{R}$ and $MinPts, \lambda \in \mathbb{N}$. A point $o \in DB$ is called correlation core point *w.r.t. ε, $MinPts$, δ, and λ (denoted by $\textsc{Core}_{den}^{cor}(o)$) if its ε-neighborhood is a λ-dimensional linear correlation set and its correlation ε-neighborhood contains at least $MinPts$ points, formally:*

$$\textsc{Core}_{den}^{cor}(o) \Leftrightarrow \textsc{CorSet}_\delta^\lambda(\mathcal{N}_\varepsilon(o)) \wedge |\mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_o}(o)| \geq MinPts.$$

Let us note that in $\textsc{Core}_{den}^{cor}$ 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 9.8 (Direct correlation reachable)
Let $\varepsilon, \delta \in \mathbb{R}$ and $MinPts, \lambda \in \mathbb{N}$. 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 $\textsc{DirReach}_{den}^{cor}(q,p)$) if q is a correlation core point, the correlation dimension of $\mathcal{N}_\varepsilon(p)$ is at most λ, and $p \in \mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_q}(q)$, formally:*

$\textsc{DirReach}_{den}^{cor}(q,p) \Leftrightarrow$

(1) $\textsc{Core}_{den}^{cor}(q)$

(2) $\textsc{CorDim}(\mathcal{N}_\varepsilon(p)) \leq \lambda$

(3) $p \in \mathcal{N}_{\varepsilon}^{\hat{\mathbf{M}}_q}(q)$.

Direct correlation reachability is symmetric only for pairs of correlation core points. Both points p and q must find the other point in their corresponding correlation ε-neighborhood.

Definition 9.9 (correlation reachable)
Let $\varepsilon, \delta \in \mathbb{R}$ ($\delta \approx 0$) and MinPts, $\lambda \in \mathbb{N}$. A point $p \in \mathcal{D}$ is correlation reachable *from a point $q \in \mathcal{D}$ w.r.t. ε, MinPts, δ, and λ (denoted by $\text{REACH}_{den}^{cor}(q,p)$) if there is a chain of points $p_1, \cdots p_n$ such that $p_1 = q, p_n = p$ and p_{i+1} is direct correlation reachable from p_i, formally:*

$$\begin{aligned}&\text{REACH}_{den}^{cor}(q,p) \Leftrightarrow \\ &\quad \exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = q \ \wedge \ p_n = p \ \wedge \\ &\quad \forall i \in \{1, \ldots, n-1\} : \text{DIRREACH}_{den}^{cor}(p_i, p_{i+1}).\end{aligned}$$

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

Definition 9.10 (correlation connected)
Let $\varepsilon, \delta \in \mathbb{R}$ ($\delta \approx 0$) and MinPts, $\lambda \in \mathbb{N}$. A point $p \in \mathcal{D}$ is correlation connected *to a point $q \in \mathcal{D}$ w.r.t. ε, MinPts, δ, and λ (denoted by $\text{CONNECT}_{den}^{cor}(q,p)$) if there is a point $o \in \mathcal{D}$ such that both p and q are correlation reachable from o, formally:*

$$\begin{aligned}&\text{CONNECT}_{den}^{corr}(q,p) \Leftrightarrow \\ &\exists o \in \mathcal{D} : \text{REACH}_{den}^{corr}(o,q) \ \wedge \ \text{REACH}_{den}^{corr}(o,p).\end{aligned}$$

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

Definition 9.11 (correlation connected cluster)
Let $\varepsilon, \delta \in \mathbb{R}$ ($\delta \approx 0$) and MinPts, $\lambda \in \mathbb{N}$. A non-empty subset $C \subseteq \mathcal{D}$

is called a correlation connected cluster *w.r.t. ε, MinPts, δ, and λ if all points in C are correlation connected and C is maximal w.r.t. correlation reachability, formally:*

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

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

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

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, analog to DBSCAN. First, choose an arbitrary correlation core point o from the database. Second, retrieve all points that are correlation reachable from o. This approach yields the correlation connected cluster containing o.

Lemma 9.2

Let $p \in \mathcal{D}$. If p is a correlation core point, then the set of points which are correlation reachable from p is a correlation connected cluster, formally:

$$\textsc{Core}_{den}^{cor}(p) \wedge C = \{o \in \mathcal{D} \mid \textsc{Reach}_{den}^{cor}(p, o)\} \Rightarrow \textsc{Cluster}_{den}^{cor}(C).$$

Proof.

(1) $C \neq \emptyset$:

By assumption, $\textsc{Core}_{den}^{cor}(p)$ *and thus,* $\textsc{CorDim}(\mathcal{N}_{\varepsilon}^{\hat{\mathbf{M}}}(p)) \leq \lambda$.

$\Rightarrow \textsc{DirReach}_{den}^{cor}(p, p)$

$\Rightarrow \textsc{Reach}_{den}^{cor}(p, p)$

$\Rightarrow p \in C$.

(2) Maximality:

Let $x \in C$ 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).*

$\Rightarrow y \in C$.

(3) Connectivity:

$\forall x, y \in C : \textsc{Reach}_{den}^{cor}(p, x) \wedge \textsc{Reach}_{den}^{cor}(p, y)$

$\Rightarrow \textsc{Connect}_{den}^{cor}(x, y)$ *(via p).* □

Lemma 9.3

Let $C \subseteq \mathcal{D}$ be a correlation connected cluster. Let $p \in C$ be a correlation core point. Then C equals the set of points which are correlation reachable from p, formally:

$$\textsc{Cluster}_{den}^{cor}(C) \wedge p \in C \wedge \textsc{Core}_{den}^{cor}(p) \Rightarrow C = \{o \in \mathcal{D} \mid \textsc{Reach}_{den}^{cor}(p, o)\}.$$

Proof.

Let $\bar{C} = \{o \in \mathcal{D} \mid \textsc{Reach}_{den}^{cor}(p, o)\}$. We have to show that $\bar{C} = C$:

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

(2) $C \subseteq \bar{C}$: Let $q \in C$. By assumption, $p \in C$ and $\textsc{Cluster}_{den}^{cor}(C)$.
$\Rightarrow \exists o \in C : \textsc{Reach}_{den}^{cor}(o, p) \wedge \textsc{Reach}_{den}^{cor}(o, q)$
$\Rightarrow \textsc{Reach}_{den}^{cor}(p, o)$ *(since both o and p are correlation core points, and correlation reachability is symmetric for correlation core points)*
$\Rightarrow \textsc{Reach}_{den}^{cor}(p, q)$ *(transitivity of correlation-reachability)*
$\Rightarrow q \in \bar{C}$. □

9.3 Computing Correlation Connected Clusters

9.3.1 Algorithm 4C

In the following, we describe the algorithm 4C (Computing Correlation Connected Clusters). 4C performs one single pass over the database to find all correlation clusters for a given parameter setting according to Lemmata 9.2 and 9.3. The pseudocode of the algorithm 4C is given in Figure 9.5. At the beginning, each point is marked as unclassified. During the run of 4C, all points are either assigned to a certain cluster identifier or marked as noise. For each point which is not yet classified, 4C checks whether this point is a correlation core point. If the point is a correlation core point, the algorithm expands the cluster belonging to this point. Otherwise the point is marked

```
algorithm 4C(𝒟, ε, MinPts, λ, δ)
    // assumption: each point in 𝒟 is marked as unclassified
  for each unclassified o ∈ 𝒟 do
    if CORE^cor_den(o) then
      generate new clusterID;
      insert all x ∈ N_ε^M̂o(o) into queue Φ;
      while Φ ≠ ∅ do
        q = first point in Φ;
        compute ℛ = {x ∈ 𝒟 | DIRREACH^cor_den(q, x)};
        for each x ∈ ℛ do
          if x is unclassified or noise then
            assign current clusterID to x
          end if
          if x is unclassified then
            insert x into Φ;
          end if
        end for
        remove q from Φ;
      end while
    else
      mark o as noise;
    end if
  end for
```

Figure 9.5: Pseudo code of the 4C algorithm.

as noise. To find a new cluster, 4C starts with an arbitrary correlation core point o and expands a cluster by searching for all points that are correlation reachable from o. This is sufficient to find the whole cluster containing the point o, due to Lemmata 9.2 and 9.3. When 4C finds a new initial correlation core point, a new cluster identifier clusterID is generated which will be assigned to all points found during the expansion. 4C begins this expansion by inserting all points in the correlation ε-neighborhood of point o into a queue. For each point in the queue, it computes all directly correlation reachable points and inserts those points into the queue which are still unclassified. This is repeated until the queue is empty.

Obviously, the results of 4C do not depend on the order of processing, i.e. the resulting clustering (number of clusters and association of core points to clusters) is determinate.

9.3.2 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 dimensionality. The idea of our correlation connected clustering method is founded on DBSCAN. 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 $O(n \log n)$ [EKSX96]. 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 $O(d^2 \cdot n \log n)$. ORCLUS claims to have a runtime complexity of $O(k^3 + k \cdot n \cdot d + k^2 \cdot d^3)$ where k is the number of clusters required as input parameter [AY00].

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

Lemma 9.4 *The overall worst-case time complexity of our algorithm on top of the sequential scan of the data set is $O(d^2 \cdot n^2 + d^3 \cdot n)$.*

Proof. *Our algorithm has to associate each point of the data set with a similarity measure that is used for searching neighbors (cf. Definition 9.4). 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 9.7, 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 a 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 [BKK96, BBJ+00] 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 given as follows:

Lemma 9.5 *The overall worst case time complexity of our algorithm on top of an efficient index structure for high dimensional data is* $O(d^2 \cdot n \log n + d^3 \cdot n)$.

Proof. *Analogous to Lemma 9.4.* □

9.3.3 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 points which contribute to the covariance matrix and consequently to the correlation similarity measure of each point. It also participates in the determination of the density threshold, a correlation 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 insufficiently small number of points contribute to the correlation similarity measure of each point and thus, this measure can be meaningless. On the other hand, ε should not be too large because then some noise points might be correlation reachable from points 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 a point 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 ($MinPts \geq 5$ 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. Only those clusters with a correlation dimensionality of no more than λ are determined. 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. If the correlation dimensionality of the clusters is unknown, 4C must simply be started with several selections of λ, since clusters of different correlation dimensionality may form a hierarchy (e.g. two 2-dimensional clusters may together form a 3-dimensional cluster).

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 $\kappa \gg 1$. It empirically turned out that 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.

9.4 Quality Evaluation

In this section, we present a broad efficiency evaluation of 4C. The evaluation is based on several synthetic data sets as well as on the Tavazoie gene expression data and the Metabolome data (cf. Section 7.3). In addition, we compared the quality of the results of our method to the quality of the

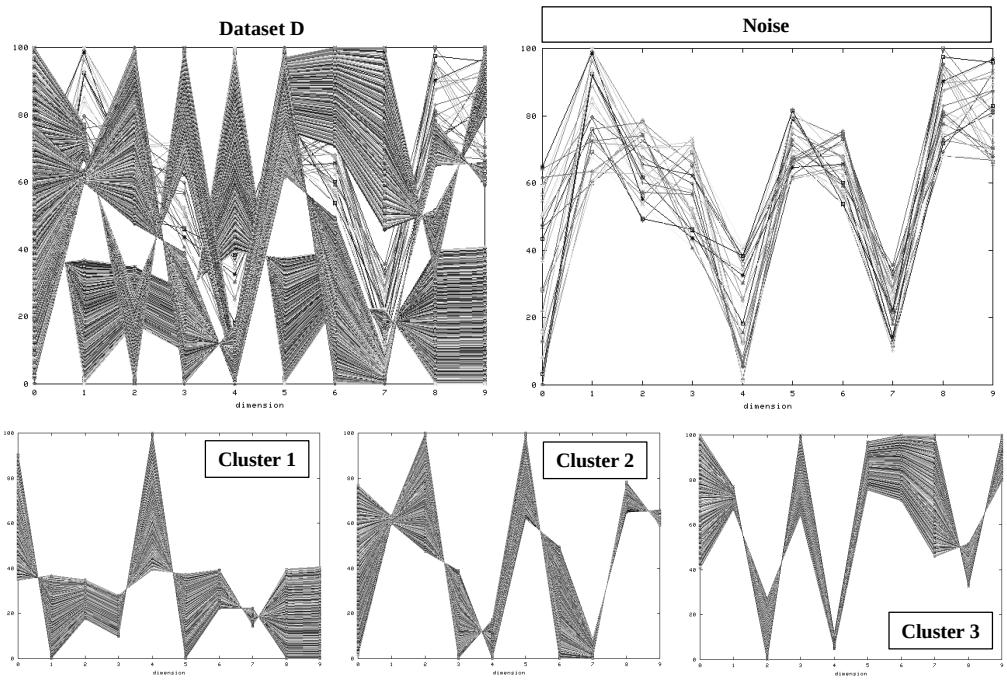


Figure 9.6: Transposed view of three clusters and noise found by 4C on a 10D synthetic data set. Parameters: $\varepsilon = 10.0$, $MinPts = 5$, $\lambda = 2$, $\delta = 0.1$.

results of DBSCAN, ORCLUS, and CLIQUE. In all our experiments, we set the parameter $\kappa = 50$ as suggested in Section 9.2.2.

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 separate the correlation-connected clusters from noise. As an example, Figure 9.6 illustrates the transposed view of the three clusters and the noise 4C found on a sample 10-dimensional synthetic data set consisting of approximately 1,000 points. Applied to all synthetic data sets, 4C computed 100% accuracy.

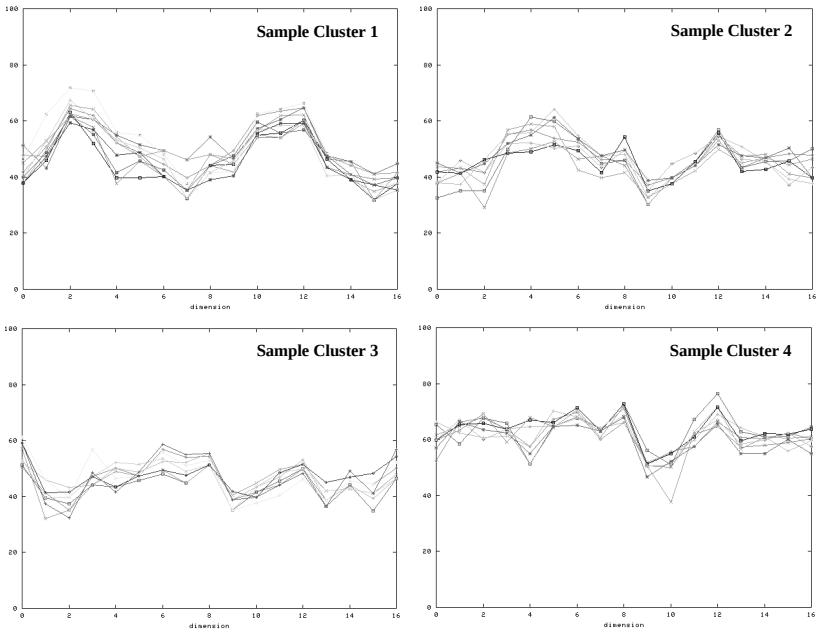


Figure 9.7: Sample clusters found by 4C on the gene expression data set. Parameters: $\varepsilon = 25.0$, $MinPts = 8$, $\lambda = 8$, $\delta = 0.01$.

Real-World Data Sets

Gene Expression Data. 4C found 60 correlation connected clusters of co-regulated genes (10-20). Such small cluster sizes are quite reasonable from a biological perspective. The transposed views of four sample clusters are depicted in Figure 9.7. 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. We also analyzed the results of our correlation clusters and found several biologically important implications. For example, one cluster consists of several genes coding for proteins related to the assembly of the spindle pole, required for mitosis (e.g. KIP1, SLI15, SPC110, SPC25, and NUD1). Another cluster contains several genes coding for structural constituents of the ribosome (e.g. RPL4B, RPL15A, RPL17B, and RPL39). The functional relationships of the genes in the clusters confirm the significance of the computed co-regulation.

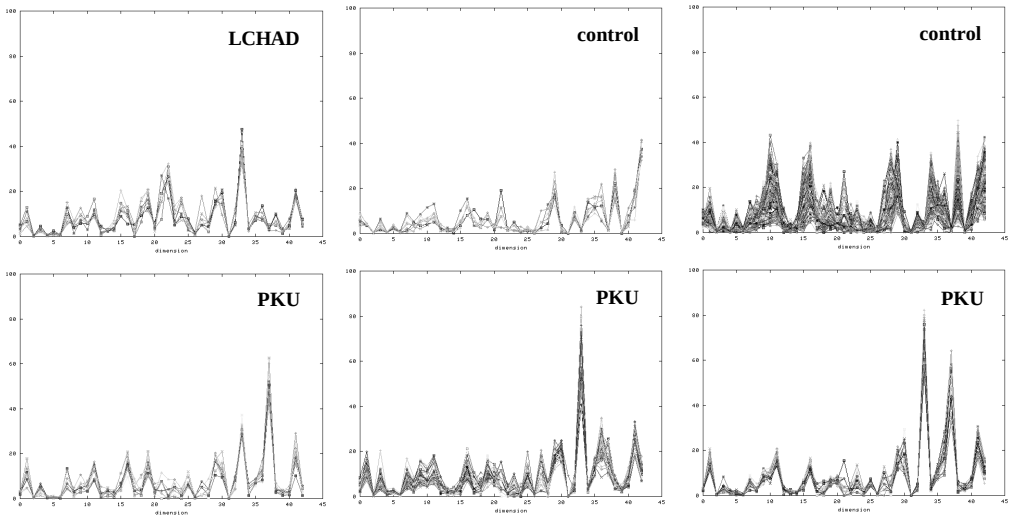


Figure 9.8: Clusters found by 4C on the metabolome data set. Parameters: $\varepsilon = 150.0$, $MinPts = 8$, $\lambda = 20$, $\delta = 0.1$.

Metabolome Data. 4C detected six correlation connected sets which are visualized in Figure 9.8. Cluster one and two (in the upper right 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. Let us note that the computed clusters are 100% pure, i.e. they only contain instances of a single class.

Comparisons to Other Methods

We compared the effectiveness of 4C with related clustering methods, in particular the density-based clustering algorithm DBSCAN, the subspace clustering algorithm CLIQUE, 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 ($d = 10$).

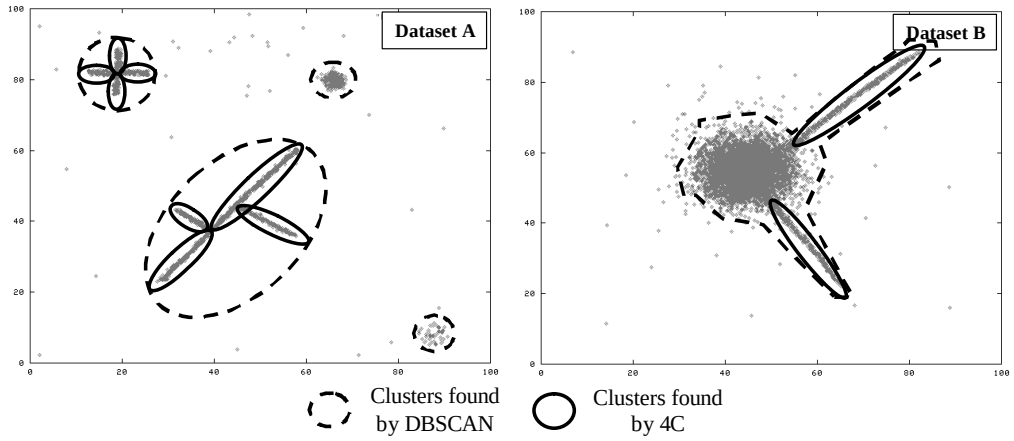


Figure 9.9: Comparison between 4C and DBSCAN.

Comparison with DBSCAN. The clusters found by DBSCAN and 4C applied on the 2-dimensional data sets are depicted in Figure 9.9. 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 9.9) 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 CLIQUE. A comparison of 4C with CLIQUE gained similar results. CLIQUE finds clusters in subspaces which do not exhibit correlations (and thus are not detected by 4C). On the other hand, CLIQUE is usually limited to axis-parallel clusters and therefore cannot detect arbitrary correlations. These observations occur especially with higher dimensional data ($d \geq 10$ in our tests). Again, these results are not astonishing since CLIQUE only searches for axis-parallel subspace clusters (dense projections) but does not search for correlations. This empirically supported the suspicion that CLIQUE cannot be applied to the task of finding correlation connected

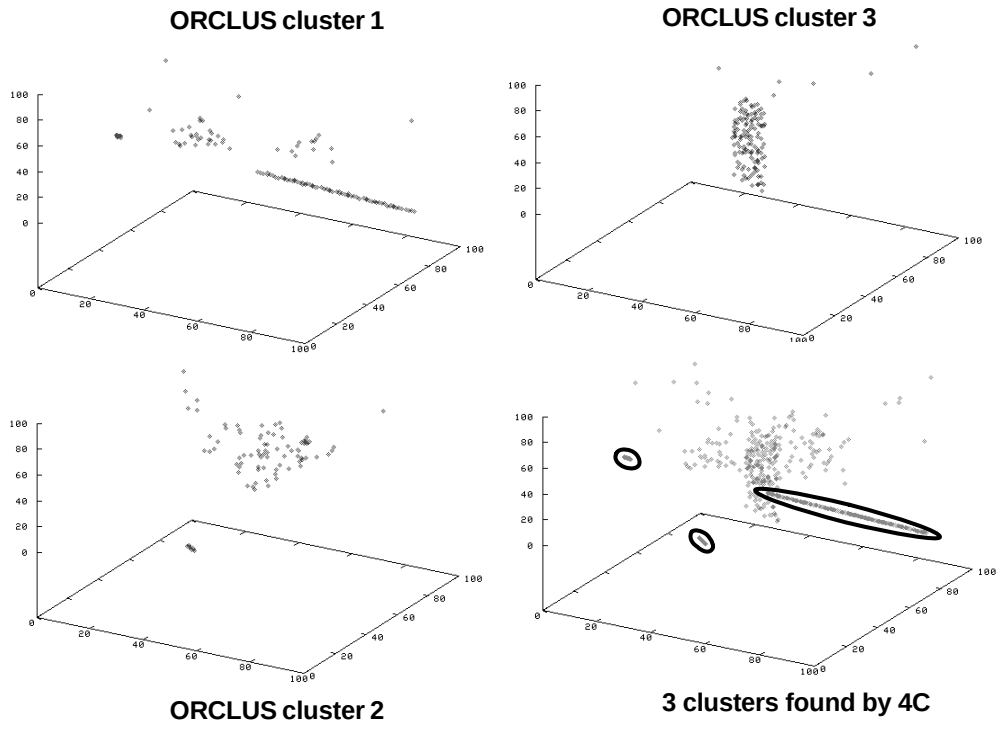


Figure 9.10: Clusters found by 4C (parameters: $\varepsilon = 2.5$, $MinPts = 8$, $\delta = 0.1$, $\lambda = 2$), and ORCLUS (parameters: $k = 3$, $l = 2$).

sets. In general, subspace clustering and correlation clustering algorithms aim at different results.

Comparison with ORCLUS. A comparison of 4C with ORCLUS resulted in quite different observations. In fact, ORCLUS computes clusters of correlated points. However, since it is a k-means 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 points 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 points and thus are

hard to obtain from the resulting output. Figure 9.10 illustrates a sample 3-dimensional synthetic data set, the clusters found by 4C are marked by black lines. Figure 9.10 depicts the points 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 points. When we applied ORCLUS on higher dimensional data sets ($d = 10$), the choice of k became even more complex and the problem of noise points 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.

9.5 Modifications and Specializations

In this section, we will propose two small variations of the concepts underlying 4C. One variation is a modification to identify pattern-based clusters, the second is a specialization to address the projected clustering approach.

9.5.1 A Variant for Pattern-Based Clustering

General Idea

4C computes arbitrary linear correlations that exhibit a given density. However, for pattern-based clustering, the density constraint should be relaxed. Intuitively, we want to add all points that are located on a correlation hyperplane and not only those that are also dense. 4C provides a solid basis to achieve this claim. The key idea is to modify the similarity matrix of a point in Definition 9.4 in the following way. Instead of setting the eigenvalues of $\hat{\mathbf{E}}_p$ to 1 and κ, we set it to 0 and κ. The resulting similarity matrix is called *pattern-based* similarity matrix instead of *correlation* similarity matrix.

Definition 9.12 (pattern-based similarity matrix of a point)
Let $p \in \mathcal{D}$ and $\mathbf{V}_p$, $\mathbf{E}_p$ 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

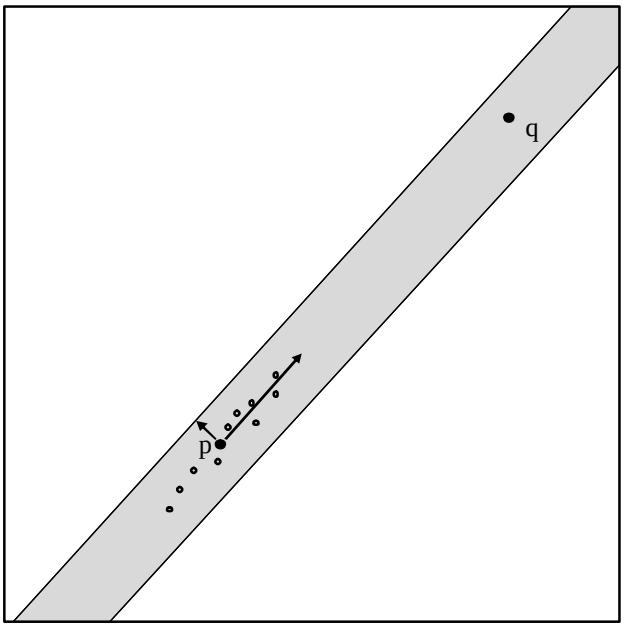


Figure 9.11: Visualization of the adopted correlation similarity measure for pattern-based clustering.

$\breve{\mathbf{E}}_p$ *with entries* $\breve{e}_i$ $(i = 1, \ldots d)$ *is computed from the eigenvalues* $e_1, \ldots, e_d$ *in* $\mathbf{E}_p$ *according to the following rule:*

$$\breve{e}_i = \begin{cases} 0 & if \quad \Omega(e_i) > \delta \\ \kappa & if \quad \Omega(e_i) \leq \delta \end{cases}$$

where Ω *is the already known normalization of the eigenvalues onto* $[0, 1]$. *The matrix* $\breve{\mathbf{M}}_p = \mathbf{V}_p \breve{\mathbf{E}}_p \mathbf{V}_p^{\mathbf{T}}$ *is called the* pattern-based similarity matrix of point p. *The* pattern-based similarity measure associated with point p *is denoted by*

$$dist_p^{pat}(p, q) = \sqrt{(p - q) \cdot \breve{\mathbf{M}}_p \cdot (p - q)^{\mathbf{T}}}.$$

All other concepts known from Section 9.2.2 remain unchanged, i.e. we can define a general (symmetric) *pattern-based* distance function very similar to Definition 9.5 and we can also define a *pattern-based* ε-neighborhood, *pattern-based* core points, *pattern-based* direct reachability, etc. The adopted algorithm based on this notion of *pattern connected clusters* works in principle like 4C but detects pattern-based clusters.

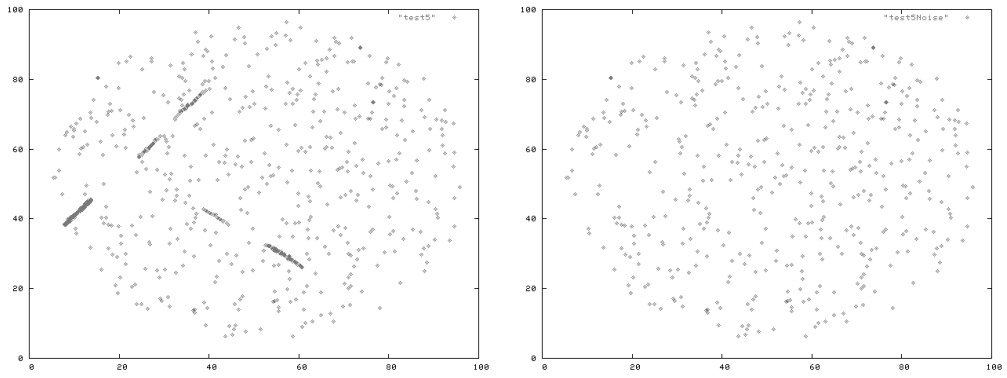


Figure 9.12: Synthetic test data set (left) and points classified as noise by the pattern-based variant of 4C (right).

The effect of the modified pattern-based similarity measure is visualized in Figure 9.11. The adopted pattern-based ε-neighborhood captures the complete data space along the direction of highest variance, whereas its extension along the direction of lowest variation remains ε/κ. As a consequence, points that are correlated but are not dense are added to a common cluster, e.g. p and q in Figure 9.11. The adoption of 4C based on that notion of pattern connected clusters is more general than the algorithmic schemes proposed for pattern-based clustering so far. However, both points p and q in Figure 9.11 must still exhibit a similar correlation in their local neighborhood because otherwise, they may have different similarity measures and may not find each other in their according adopted correlation ε-neighborhood.

Experimental Results

We tested the pattern-based variant of 4C, using several synthetic data sets, that contained one or more lower dimensional pattern-based clusters. The results of a sample 2-dimensional data set (cf. Figure 9.12) is depicted in Figure 9.13. As it can be seen from the transposed view, the clusters contain all points that exhibit a similar pattern. In addition, the points in the cluster need not to be dense anymore. However, as it can be seen from the results,

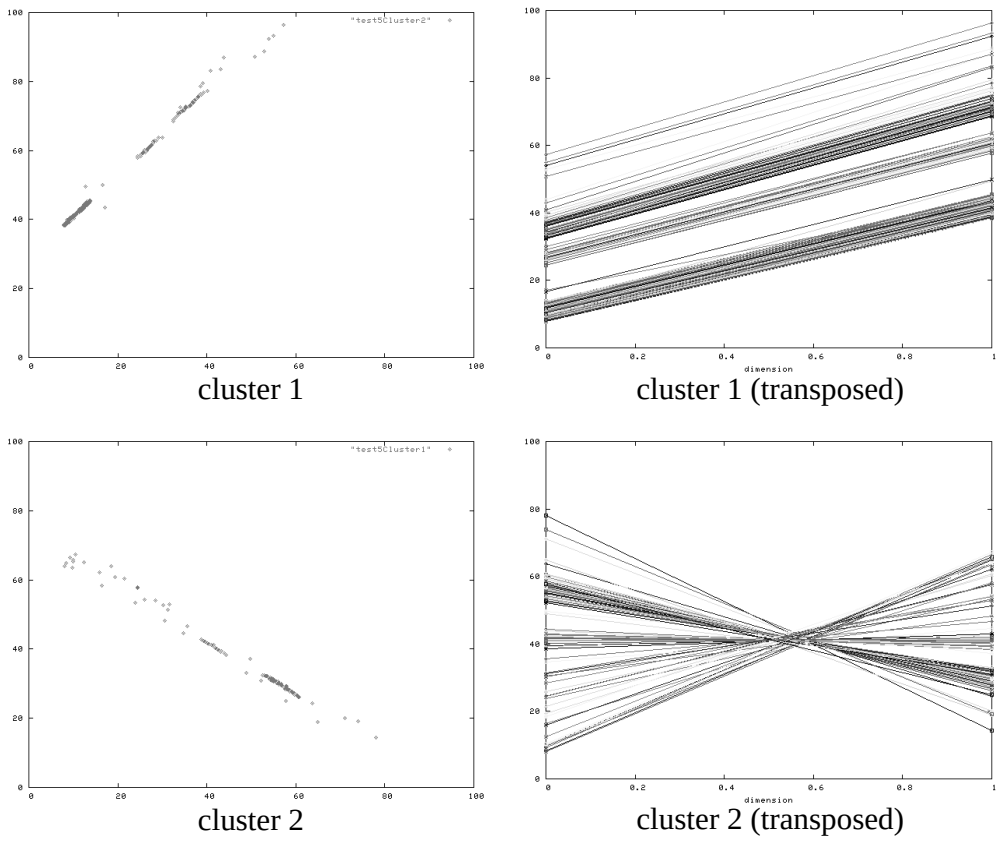


Figure 9.13: Clusters found by the pattern-based variant of 4C.

we still need local density to detect any pattern-based cluster.

9.5.2 A Variant for Projected Clustering

General Idea

4C detects arbitrarily oriented dense hyperplanes, i.e. projections where the points exhibit a certain density and that are arbitrarily oriented in the feature space. On the other hand, projected clustering aims at detecting axis-parallel dense projections. Based on the modification for the pattern-based clustering approach in the previous subsection, a density-based projected clustering algorithm is in sight which is able to detect projected clusters of

arbitrary shape and size. This would be an enhancement of existing methods such as PROCLUS [APW+99] which is k-means based or DOC (cf. Chapter 8). Both approaches suffer from the fact that they cannot detect projected clusters of arbitrary shape and size. In order to compute *density connected projected clusters*, we just need to adopt the concepts of pattern connected clusters to compute *density-based projected clusters*. Instead of computing the principal axis of a set of points, we now search for low variation along one or more axes. The attributes that exhibit a low variation should be weighted by κ and the other attributes by 0. Thus, our new projected similarity measure is a weighted Euclidean distance function with weights κ and 0. The only question that remains is how to distinguish between attributes of low variation and attributes of high variation. A natural choice to decide about the variation of the points around a point p in the projection onto an attribute a_i is to test whether the ε-neighborhood of p projected onto a_i, i.e. $\mathcal{N}_\varepsilon^{\{a_i\}}(p)$ contains at least *MinPts* points. Based on these considerations, we can define the adopted *projection similarity measure of a point* as follows.

Definition 9.13 (projection similarity measure of a point)
Let $p \in \mathcal{D}$. Let $\bar{\mathbf{w}}_p$ be the so-called projection similarity weight vector of point *p:*

$$\bar{\mathbf{w}}_p = (w_1, w_2, ...w_d),$$

where

$$w_i = \begin{cases} 0 & \text{if } \mathcal{N}_\varepsilon^{\{a_i\}}(p) \leq \mathit{MinPts} \\ 1 & \text{else.} \end{cases}$$

The projected similarity measure associated with a point p *is denoted by*

$$dist_p^{proj}(p,q) = \sqrt{\sum_{i=1}^{d} w_i \cdot (\pi_{\{a_i\}}(p) - \pi_{\{a_i\}}(q))^2}$$

where w_i is the i-th component of $\bar{\mathbf{w}}_p$.

Using this projection similarity measure of a point, we can define the general symmetric extension of this measure analogously to Definition 9.5. Thus, we can define a *projected* ε-neighborhood and based on this, we can

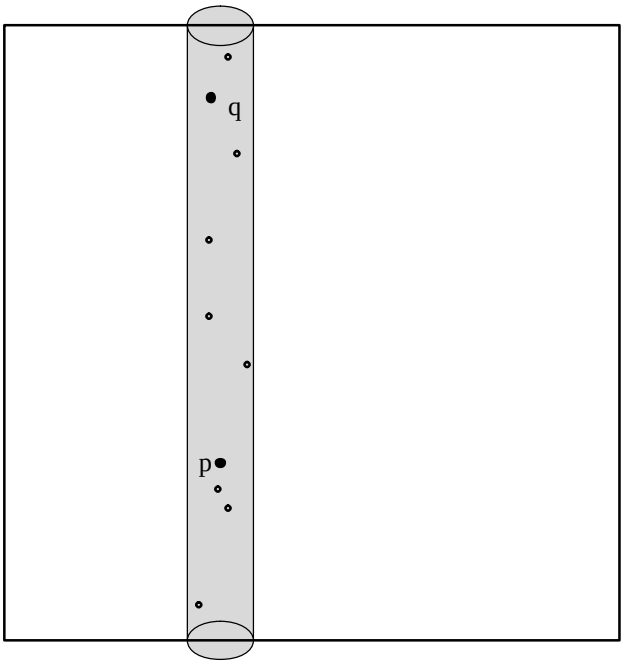
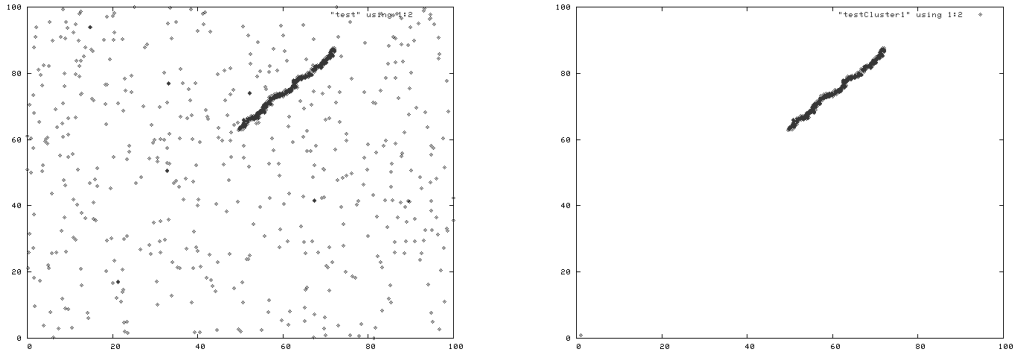


Figure 9.14: Visualization of the adopted projection similarity measure for projected clustering.

formalize the notion of *density connected projected clusters* analogously to Definitions 9.6 to 9.11.

In fact, the projected ε-neighborhoods exhibit hyper-spheres onto the according projections of low variation. Using the concept of density connectivity adopted to projected clustering as described above, we will detect clusters of arbitrarily shape and size in the according projections. The adopted projected ε-neighborhood is depicted in Figure 9.14 for a 2-dimensional point p. The projected ε-neighborhood captures the complete data space along the axes of high variance, whereas its extension along the axes of low variation is ε, i.e. it forms a hyper-sphere with radius ε onto the projection of low variation. Thus, point q in Figure 9.14 will be inserted into the projected cluster of p. In fact, the projected clusters are rather similar to those DBSCAN would find in the according projections of the data space.



(a) 2-dimensional projection of a 50-dimensional synthetic data set

(b) 2-dimensional projection of the cluster found by the adoption of 4C

Figure 9.15: Results of the projected clustering variation of 4C on a sample synthetic data set

Experimental Results

We tested the adoption of 4C to projected clustering using several synthetic data sets that contained one or more lower dimensional projected clusters of arbitrary shape and size. The data were generated to guarantee that DBSCAN cannot find any cluster in full-dimensional space. A sample result is depicted in Figure 9.15. The 2-dimensional projection of a 50-dimensional synthetic data set, in which there is a cluster, is depicted in Figure 9.15(a). The data set contains one 2-dimensional projected cluster of complex shape. Our adoption of 4C detected this cluster by an accuracy of 100% as can be seen in Figure 9.15(b). Our further tests on other synthetic data sets confirm these results.

9.6 Summary

In this Chapter, we proposed a density-based approach to find sets of linearly correlated, densely packed points in a high dimensional feature space. Our formal notion of *correlation connected clusters* combines the (full-dimensional) density connected notion of clusters with the concept of PCA. We devel-

oped an algorithm called 4C (Computing Correlation Connected Clusters) for efficiently detecting such correlation connected clusters. Due to the well-founded clustering notion, the correctness of 4C can be formally proven. 4C outperforms existing correlation clustering algorithms (especially ORCLUS) in terms of accuracy because it is not sensitive to noise and can detect any linear correlation with a dimensionality lower than the user defined threshold λ. In addition, the results of 4C do not depend on the order of processing, and the assignment of points to clusters or noise is determinate (at least for correlation core points). A broad experimental evaluation on synthetic and real-world data sets including the Metabolome data set and the Tavazoie gene expression data set empirically confirmed this proper performance of 4C.

In addition, we presented two extensions to the concept of 4C. The first extension addresses the pattern-based clustering approach. Using this extension, 4C detects clusters of points that exhibit an arbitrary linear tendency (pattern) in a subset of their attributes. This is a generalization of current pattern-based approaches that are limited to find clusters of points showing only less complex tendencies. The second extension addresses the projected clustering approach. Using this extension, 4C detects projected clusters of arbitrary shape and size. The advantage of a determinate result is received. This is an enhancement to existing projected clustering methods (e.g. PROCLUS and DOC) that suffer mainly from non-determine results and rather simple clustering models that favors particular cluster shapes. The accuracy of both extensions where illustrated using synthetic data sets.

Part IV

Conclusions

Chapter 10

Summary and Future Directions

Within the KDD process, data mining is the application of algorithms to discover patterns and trends in large databases. Clustering is one of the most important data mining tasks. The methods and concepts presented in this thesis contribute to the solution of novel challenges for clustering algorithms. This chapter summerizes the main contributions of this thesis (Section 10.1) and shows potentials for future research directions (Section 10.2).

10.1 Summary of Contributions

The rapidly increasing amount of data stored in databases requires efficient and effective data mining methods to make the full use out of the collected data. Clustering is one of the primary data mining tasks and aims at detecting subgroups of similar data objects. This thesis contributes in the field of clustering. New and original solutions for novel challenges of clustering algorithms, in particular for the density-based clustering approach, which is one of the most successful clustering models, are proposed. In the following, we give a detailed summary of these contributions.

10.1.1 Preliminaries (Part I)

The preliminaries in Part I illustrate the topic and the background of this work. After a very general introduction to KDD, data mining, and clustering, we give a classification of general clustering algorithms proposed recently. The density-based clustering notion underlying the algorithms DBSCAN and OPTICS which forms the basis of this thesis is reviewed in more detail. In addition, basic notations are introduced.

10.1.2 Using Density-Based Hierarchical Clustering for Similarity Search Applications (Part II)

Part II presents an industrial prototype called BOSS (Browsing OPTICS Plots for Similarity Search) that enables visual data browsing based on a hierarchy of clusters computed by OPTICS. BOSS is a first step towards developing a comprehensive and scalable solution, designed to make the efficiency and the analytical potentials of OPTICS available to a broader audience. In particular, BOSS is designed to support the following important application ranges:

- **Visual Data Mining:** BOSS enables to visually browse the results of the cluster hierarchy generated by OPTICS. This supports the user in

analyzing the clustering results, i.e. a semi-automatic cluster analysis of massive data sets.

- **Interactive Similarity Search:** BOSS enables a user to visually search for similar data objects without the requirement to specify or sketch a query object. Applied to CAD databases and digital engineering (e.g. of car parts), BOSS can provide engineers with a quick overview of already existing data objects (i.e. parts). Engineers are able to navigate their way through the diversity of existing variants of products and parts, reducing the costs of developing and producing new parts by maximizing the reuse of existing parts.

- **Evaluation of Similarity Models:** Effective similarity models form the basis of accurate similarity search. In general, similarity models can be evaluated by computing sample similarity queries. However, this procedure is subjective and error-prone, since the quality of a model depends on the results of few sample queries and, therefore, on the choice of the query objects. Applying a clustering algorithm is much more objective, taking all data objects into account for evaluation. BOSS helps to analyze how accurate the clustering structure generated by OPTICS reflects the intuitive notion of similarity. This helps to decide about the accuracy of the similarity model.

In Part II, we identify three key requirements for the development of BOSS that have been insufficiently addressed so far by other approaches or not yet addressed at all. These requirements include an incremental version of OPTICS to cope with large dynamic (i.e. frequently updated) databases, solid cluster extraction from hierarchical cluster representations, and selection of meaningful cluster representatives. The three requirements are improvements of the density-based hierarchical clustering method OPTICS. We present solutions for these requirements in this work.

First, we propose an incremental variant of OPTICS called IncOPTICS for incrementally maintaining the clustering structure after the insertion or deletion of an update object. The basic algorithms are extended for handling

bulk updates. IncOPTICS achieves significant speed-up factors over OPTICS and thus enables the application of BOSS in a dynamic environment.

Second, we present a novel algorithm called GradientClustering for extracting clusters from hierarchical representations generated by OPTICS. The GradientClustering algorithm is designed to meet the requirements of BOSS, especially to compute a cluster hierarchy of deep details. It outperforms recent comparative approaches in terms of these requirements. In addition, we proposed two approaches for selecting meaningful cluster representatives based on the density-based concepts underlying OPTICS. Both approaches have shown to produce better results than the well-known medoid approach.

We illustrate some details on the implementation of the BOSS prototype, incooperating the ideas presented in Part II. In addition, we outline two sample applications, first, to visual data mining and semi-automatic cluster analysis in a database of protein structures and second, to the evaluation of similarity models using a database of car parts. Both examples show the sound usability of BOSS.

10.1.3 Adopting Density-Based Clustering to High Dimensional Data (Part III)

Part III deals with the problem of high dimensional feature databases which is an active area of research. We first give a general introduction to the problems of clustering high dimensional data, summarized by the term *curse of dimensionality*. After that, we classify current approaches for clustering high dimensional data into projected clustering, subspace clustering, pattern-based clustering, and correlation clustering algorithms. Each class of approaches has different aims and, therefore, different requirements. Density-based clustering is then combined with dedicated approaches to deal with that special requirements.

A density-based subspace clustering algorithm called SUBCLU (density-based Subspace Clustering) is proposed. It automatically and efficiently computes all "flat" subspace clusters DBSCAN would have found if applied to all

possible subspaces. SUBCLU is applied to a real-world gene expression data set outperforming comparative subspace clustering approaches and yielding a significant amount of important biological information. A second algorithm called RIS (Ranking Interesting Subspaces), a semi-hierarchical extension of SUBCLU, is proposed for the subspace clustering problem. The main difference to SUBCLU is that RIS ranks the subspaces according to their clustering quality rather than computing subspace clusters. A user can choose some subspaces from a list sorted by clustering quality and apply his/her own (e.g. hierarchical) clustering algorithm to the particular subspaces. The advantage of RIS is that it can be combined with a hierarchical clustering algorithm. The combination of RIS and OPTICS is applied to gene expression data, yielding further important insights that were missed by SUBCLU.

In addition, we combined the density-based clustering notion with PCA, a primitive to measure correlation. Based on this combination, a sound formalization of *correlation connected clusters* is presented. We propose an efficient algorithm called 4C to compute such correlation connected clusters and apply this method on a gene expression data set and on a metabolome data set. 4C shows a significant accuracy gain compared to other clustering methods. In addition, two extensions of 4C are presented. One extension aims at finding pattern-based clusters and the second extension is able to compute density-based projected clusters of arbitrary size and shape.

In summary, we applied the density-based clustering notion to the approaches for clustering high dimensional data. The benefit of the proposed methods is that the advantages of this powerful clustering model are conserved.

10.2 Future Work

At the end of this thesis, let us emphasize the potentials of the proposed methods for clustering.

For BOSS, we see the following opportunities for future research:

- To improve the representatives displayed in the browsable hierarchy, a quality measure is needed. Such a quality measure for cluster representatives could be based on the concept of local outlier detection, determining how strong a point is an outlier w.r.t. the other objects in the cluster. Having such a quality measure at hand, we could compare the representatives generated by the different approaches incooperated within BOSS and could e.g. present a ranked list of representatives to the user.

- In many real-world databases, the data objects are distributed over several sites. A parallel and/or distributed version of OPTICS may be required since a centralized clustering could be impossible due to network bandwidth constraints. This would be the first step towards a BOSS system for a distributed database environment.

For the clustering of high dimensional data, future research could be guided in the following directions:

- Currently, subspace clustering/ranking algorithms are limited by the use of a global density threshold. The development of a density-based subspace ranking method which is adoptable to local density would be an enhancement to SUBCLU and RIS. However, it is not clear how the concepts of OPTICS can be adopted for efficient subspace clustering.

- Beside the approach of inverted files, there are no index structures for partial range queries, i.e. range queries in arbitrary subspaces of the feature space, needed by the SUBCLU and RIS algorithms. An open question is, if traditional index structures, which originally cannot be applied to this problem, can be adopted to support partial range queries more efficiently.

- Computing hierarchies of correlation connected clusters is another open question. Currently, 4C can only detect correlations of a fixed correlation dimension. However, two k-dimensional correlations can e.g. form a $(k+1)$-dimensional correlation. It would be interesting to investigate

how the concepts of correlation connected clusters could be extended to find correlation hierarchies.

- Last but not least, it is interesting to combine the density-based clustering notion with other correlation primitives. Beside PCA (used in 4C as proposed in this thesis), there are several other concepts such as fractal dimension, Hough transformations, etc. which could be used. It could even be interesting to design a general framework where the user can combine the density-based clustering notion with primitives for correlation analysis of his/her choice.

List of Figures

1.1 The KDD process. 5

2.1 A dendrogram (right) for a sample data set (left). 16
2.2 The idea of feature transformation. 18
2.3 Sample databases. 19
2.4 Illustration of density-based clustering concepts 22
2.5 The DBSCAN algorithm. 23
2.6 Method ExpandCluster. 24
2.7 Nested clusters of different density. 25
2.8 Illustration of core distance and reachability distance. 27
2.9 The OPTICS algorithm. 28
2.10 Method OrderedSeeds::update. 29
2.11 Reachability plot (right) computed by OPTICS for a sample 2-D data set (left). 29

3.1 Different approaches to visual data mining [Ank00]. 36
3.2 Browsing through reachability plots. 37
3.3 Hierarchically ordered representatives. 38

4.1 Sample dataset where the entire cluster ordering is affected by the insertion/deletion of point U. 47
4.2 The core distance of q changes due to insertion/deletion of p. . 50
4.3 Algorithm insert for IncOPTICS. 55

4.4 IncOPTICS: adopted method OrderedSeeds::update. 56
4.5 Algorithm delete for IncOPTICS. 59
4.6 IncOPTICS: method OrderedSeeds::updateAll. 60
4.7 Runtime speed-up factors of IncOPTICS vs. OPTICS. 62
4.8 Results of IncOPTICS on synthetic and real-world TV data. . 63
4.9 Comparison of bulk IncOPTICS vs. OPTICS. 64

5.1 Sample nested clusters: data space (left); reachability plot (middle); cluster hierarchy (right) 69
5.2 Gradient vectors $\vec{g}(x, y)$ and $\vec{g}(y, z)$ of objects x, y and z adjacent in the cluster ordering. 71
5.3 Illustration of inflection points measuring the angle between the gradient vectors of objects adjacent in the ordering. 73
5.4 Clusters found on car parts and proteins by: a) GradientClustering, b) ξ-Clustering, c) `cluster_tree` 75
5.5 Representing clusters by superimposing all contained objects. . 78
5.6 Illustration of the minimum core distance approach. 79
5.7 Sample successor graph for a cluster of seven objects. 80
5.8 Illustration of the maximum successor approach. 82
5.9 A cluster of CAD objects with corresponding representative objects. 83
5.10 A cluster of proteins with corresponding representative objects. 84
5.11 Pseudo code of the GradientClustering algorithm. 86

6.1 BOSS distributed architecture. 88
6.2 BOSS screenshot. 89
6.3 OPTICS plot of the protein data set. 90
6.4 Sample cluster 1 found on the protein database. 91
6.5 Sample cluster 2 found on the protein database. 92
6.6 Reachability plots computed by OPTICS using different similarity models. 93

6.7 Contents of the clusters detected in Figure 6.6. 94

7.1 Probability of a point near by the data space boundary. 101
7.2 Sample projected clusters in different subspaces. 103
7.3 Sample objects cluster differently in varying subspaces. 104
7.4 Transposed view (left) and pattern-based cluster (right) of some sample database objects. 105
7.5 A 2-dimensional correlation plane in a 3-dimensional feature space. 106
7.6 Gene expression data matrix: schematic view (left), visualization of a sample raw data excerpt (right). 108

8.1 Illustration of drawbacks of existing subspace clustering algorithms. 116
8.2 Monotonicity of density connected (the circles indicate the ε-neighborhoods, $MinPts = 4$). 123
8.3 Visualization of a density connected cluster C loosing its maximality w.r.t. density reachability in a subspace. 125
8.4 The SUBCLU algorithm. 127
8.5 Procedure GenerateCandidates. 128
8.6 Scalability of SUBCLU. 130
8.7 Problems with a global density parameter. 135
8.8 The RIS algorithm. 140
8.9 Illustration of the periodic extension of the data space ($dist = L_\infty$). 142
8.10 Efficiency evaluation. 145
8.11 Part of the reachability plot generated by OPTICS in the subspace, ranked second by RIS. 147

9.1 1-dimensional correlation lines. 153
9.2 2-dimensional correlation planes. 155

9.3 Correlation ε-neighborhood of a point p according to (a) $\mathbf{M}_p$ and (b) $\hat{\mathbf{M}}_p$. 158
9.4 Symmetry of the correlation ε-neighborhood: (a) $p \in \mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_q}(q)$. (b) $p \notin \mathcal{N}_\varepsilon^{\hat{\mathbf{M}}_q}(q)$. 160
9.5 Pseudo code of the 4C algorithm. 165
9.6 Transposed view of three clusters and noise found by 4C on a 10D synthetic data set. Parameters: $\varepsilon = 10.0$, $MinPts = 5$, $\lambda = 2$, $\delta = 0.1$. 169
9.7 Sample clusters found by 4C on the gene expression data set. Parameters: $\varepsilon = 25.0$, $MinPts = 8$, $\lambda = 8$, $\delta = 0.01$. 170
9.8 Clusters found by 4C on the metabolome data set. Parameters: $\varepsilon = 150.0$, $MinPts = 8$, $\lambda = 20$, $\delta = 0.1$. 171
9.9 Comparison between 4C and DBSCAN. 172
9.10 Clusters found by 4C (parameters: $\varepsilon = 2.5$, $MinPts = 8$, $\delta = 0.1$, $\lambda = 2$), and ORCLUS (parameters: $k = 3$, $l = 2$). . . . 173
9.11 Visualization of the adopted correlation similarity measure for pattern-based clustering. 175
9.12 Synthetic test data set (left) and points classified as noise by the pattern-based variant of 4C (right). 176
9.13 Clusters found by the pattern-based variant of 4C. 177
9.14 Visualization of the adopted projection similarity measure for projected clustering. 179
9.15 Results of the projected clustering variation of 4C on a sample synthetic data set 180

List of Tables

5.1 CPU time for cluster recognition. 76

7.1 Summarization of gene expression data sets. 109
7.2 Class distribution of the Metabolome data set. 111

8.1 Comparative evaluation of SUBCLU and CLIQUE: Summary of the results on synthetic data sets. 131
8.2 Contents of four sample clusters in different subspaces. 132
8.3 A cluster missed by SUBCLU, but detected by RIS/OPTICS. 148

References

[ABKS99] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. "OPTICS: Ordering Points to Identify the Clustering Structure". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, pages 49–60, 1999.

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. "Efficient Similarity Search in Sequence Databases". In *Proc. 4th. Int. Conf. on Foundations of Data Organization and Algorithms (FODO'93), Evanston, ILL*, volume 730 of *Lecture Notes in Computer Science (LNCS)*, pages 69–84. Springer, 1993.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'98), Seattle, WA*, 1998.

[AHWY03] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. "A Framework for Clustering Evolving Data Streams". In *Proc. 29th Int. Conf. on Very Large Databases (VLDB'03), Berlin, Germany*, pages 81–92, 2003.

[AKKS99] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. "3D Shape Histograms for Similarity Search and Classification in Spatial Databases". In *Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99), Hong Kong, China*, volume 1651 of *Lecture Notes in Computer Science (LNCS)*, pages 207–226. Springer, 1999.

[ALSS95] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases". In *Proc. 21st Int. Conf. on Very Large Databases (VLDB'95), Zurich, Switzerland*, pages 490–501, 1995.

[Ank00] M. Ankerst. *Visual Data Mining.* PhD thesis, Institute for Computer Science, University of Munich, 2000.

[APW+99] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. "Fast Algorithms for Projected Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, 1999.

[AS94] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules". In *Proc. 20th Int. Conf. on Very Large Databases (VLDB'94), Santiago, Chile*, pages 487–499, 1994.

[AY00] C. C. Aggarwal and P. S. Yu. "Finding Generalized Projected Clusters in High Dimensional Space". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX*, 2000.

[Bar02] D. Barbara. "Requirements for Clustering Data Streams". *SIGKDD Explorations*, 3:23–27, 2002.

[BBJ+00] 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 *Proc. 16th Int. Conf. on Data Engineering (ICDE'00), San Diego, CA*, 2000.

[BBKK97] S. Berchtold, C. Böhm, D. A. Keim, and H.-P. Kriegel. "A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space". In *Proc. ACM PODS Symp. on Principles of Database Systems, Tucson, AZ*, pages 78–86, 1997.

[BJK+03] S. Brecheisen, E. Januzai, H.-P. Kriegel, P. Kröger, and M. Pfeifle. "Visual Mining of Cluster Hierarchies". In *Proc. 3rd International ICDM Workshop on Visual Data Mining (VDM@ICDM2003), Melbourne, FL*, 2003.

[BK97] S. Berchtold and H.-P. Kriegel. "S3: Similarity Search in CAD Database Systems". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'97), Tucson, AZ*, pages 564–567, 1997.

[BKK96] S. Berchtold, D. A. Keim, and H.-P. Kriegel. "The X-Tree: An Index Structure for High-Dimensional Data". In *Proc. 22nd Int. Conf. on Very Large Databases (VLDB'96), Mumbai (Bombay), India*, 1996.

[BKK+03] S. Brecheisen, H.-P. Kriegel, P. Kröger, M. Pfeifle, and M. Viermetz. "Representatives for Visually Analyzing Cluster Hierarchies". In *Proc. 4th Int. SIGKDD Workshop on Multimedia Data Mining: "Integrated Media Mining" (MDM/KDD'03), Washington, DC*, 2003.

[BKK+04] S. Brecheisen, H.-P. Kriegel, P. Kröger, M. Pfeifle, M. Viermetz, and M. Pötke. "BOSS: Browsing OPTICS-Plots for Similarity Search". In *Proc. 19th Int. Conf. on Data Engineering (ICDE'04), Boston, MA*, page 858, 2004.

[BKKP04] S. Brecheisen, H.-P. Kriegel, P. Kröger, and M. Pfeifle. "Visually Mining Through Cluster Hierarchies". In *Proc. SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, FL*, pages 400–412, 2004.

[BKKS01] M. M. Breunig, H.-P. Kriegel, P. Kröger, and J. Sander. "Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'01), Santa Barbara, CA*, pages 79–90, 2001.

[BKKZ04] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. "Computing Clusters of Correlation Connected Objects". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'04), Paris, France)*, 2004.

[Bou96] A. Bouguettaya. "On-Line Clustering". *IEEE Transactions on Knowledge and Data Engineering*, 8(2):333–339, 1996.

[BWF+00] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. "The Protein Data Bank". *Nucleic Acids Research*, 28:235–242, 2000.

[CFZ99] C.-H. Cheng, A. W.-C. Fu, and Y. Zhang. "Entropy-Based Subspace Clustering for Mining Numerical Data". In *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'99), San Diego, CA*, 1999.

[CHNW96] D. W. Cheung, J. Han, V. T. Ng, and Y Wong. "Maintenance of Discovered Association Rules in Large Databases: An Incremental Technique". In *Proc. 12nd Int. Conf. on Data Engineering (ICDE'96), New Orleans, LA*, pages 106–114, 1996.

[CHO02] C. Chen, S. Hwang, and Y. Oyang. "An Incremental Hierarchical Data Clustering Algorithm Based on Gravity Theory". In *Proc. 6th Pacific Asian Conf. on Knowledge Discovery and Data Mining (PAKDD'02) Taipei, Taiwan*, 2002.

[CKS+88] P. Cheesman, J. Kellu, M. Self, J. Stutz, W. Taylor, and D. Freeman. "AUTOCLASS: a Bayesian Classification System". In *Proc. 5th Int. Conf. on Machine Learning, Ann Arbor, MI*, pages 54–64, 1988.

[Con86] M. L. Connolly. "Shape Complementarity at the Hemoglobin a1b1 Subunit Interface". *Biopolymers*, 25:1229–1247, 1986.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces". In

Proc. 23rd Int. Conf. on Very Large Databases (VLDB'97), Athens, Greece, pages 426–435, 1997.

[Def77] D. Defays. "CLINK: An Efficient Algorithm for the Complete Link Cluster Method". *The Computer Journal*, 20(4):364–366, 1977.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B*, 39(1):1–31, 1977.

[EKS+98] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. "Incremental Clustering for Mining in a Data Warehousing Environment". In *Proc. 24th Int. Conf. on Very Large Databases (VLDB'98), New York, NY*, pages 323–333, 1998.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR*, pages 291–316, 1996.

[ESK03] L. Ertoz, M. Steinbach, and V. Kumar. "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data". In *Proc. SIAM Int. Conf. on Data Mining (SDM'03), San Francisco, CA*, 2003.

[EW98] M. Ester and R. Wittman. "Incremental Generalization for Mining in a Data Warehousing Environment". In *Proc. 6th Int. Conf. on Extending Database Technology, Valencia, Spain*, volume 1377 of *Lecture Notes in Computer Science (LNCS)*, pages 135–152. Springer, 1998.

[FAAM97] R. Feldman, Y. Aumann, A. Amir, and H. Mannila. "Efficient Algorithms for Discovering Frequent Sets in Incremental Databases". In *Proc. ACM SIGMOD Workshop on Research Issues*

on Data Mining and Knowledge Discovery, Tucson, AZ, pages 59–66, 1997.

[Fis95] D. H. Fisher. "Iterative Optimization And Simplification of Hierarchical Clusterings". In *Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining (KDD'95), Montreal, Canada*, 1995.

[FPL91] D. H. Fisher, M. J. Pazzani, and P. Langley. *Concept Formation: Knowledge and Experience in Unsupervised Learning.* Morgan Kaufmann Publishers, 1991.

[FPSS96a] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "KDD for Science Data Analysis: Issues and Examples". In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR*, pages 50–56, 1996.

[FPSS96b] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR*, pages 82–88, 1996.

[FRM94] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. "Fast Subsequence Matching in Time-Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN*, pages 419–429, 1994.

[GGR02] V. Ganti, J. Gehrke, and R. Ramakrishnan. "Mining Data Steams under Block Evolution". *SIGKDD Explorations*, 3:1–10, 2002.

[GNC99] S. Goil, H. S. Nagesh, and A. Choudhary. "MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets". Tech. Report No. CPDC-TR-9906-010, Center for Parallel and Distributed Computing, Dept. of Electrical and Computer Engineering, Northwestern University, 1999.

[HAK00] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. "What is the Nearest Neighbor in High Dimensional Spaces?". In *Proc. 26th Int. Conf. on Very Large Databases (VLDB'00), Cairo, Egypt*, pages 506–515, 2000.

[Har75] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.

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

[Jag91] H. V. Jagadish. "A Retrieval Technique for Similar Shapes". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'91), Denver, CO*, pages 208–217, 1991.

[JD88] A. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[KBK+03] H.-P. Kriegel, S. Brecheisen, P. Kröger, M. Pfeifle, and M. Schubert. "Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA)*, 2003.

[Kei99] D. A. Keim. "Efficient Geometry-based Similarity Search of 3D Spatial Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA*, pages 419–430, 1999.

[KKG03] H.-P. Kriegel, P. Kröger, and I. Gotlibovich. "Incremental OPTICS: Efficient Computation of Updates in a Hierarchical Cluster Ordering". In *5th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'03), Prague, Czech Republic*, volume 2737 of *Lecture Notes in Computer Science (LNCS)*, pages 224–233. Springer, 2003.

[KKK04] K. Kailing, H.-P. Kriegel, and P. Kröger. "Density-Connected Subspace Clustering for High-Dimensional Data". In *Proc.*

SIAM Int. Conf. on Data Mining (SDM'04), Lake Buena Vista, FL, 2004.

[KKKW03] K. Kailing, H.-P. Kriegel, P. Kröger, and S. Wanka. "Ranking Interesting Subspaces for Clustering High Dimensional Data". In *Proc. 7th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'03), Cavtat-Dubrovnic, Croatia*, volume 2838 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 241–252. Springer-Verlag, 2003.

[KKM+03] H.-P. Kriegel, P. Kröger, Z. Mashael, M. Pfeifle, M. Pötke, and T. Seidl. "Effective Similarity Search on Voxelized CAD Objects". In *Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan*, 2003.

[KKS98] G. Kastenmüller, H.-P. Kriegel, and T. Seidl. "Similarity Search in 3D Protein Databases". In *Proc. German Conf. on Bioinformatics (GCB'98), Köln, Germany*, 1998.

[KM00] F. Korn and S. Muthukrishnan. "Influenced Sets Based on Reverse Nearest Neighbor Queries". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX*, 2000.

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

[LNRvK+02] 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.

[LW03] J. Liu and W. Wang. "OP-Cluster: Clustering by Tendency in High Dimensional Spaces". In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM'03), Melbourne, FL*, 2003.

[McQ67] J. McQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In *5th Berkeley Symp. Math. Statist. Prob.*, volume 1, pages 281–297, 1967.

[MG95] R. Mehrotra and J. E. Gary. "Feature-Index-Based Similar Shape Retrieval". In *Proc. 3rd Working Conf. on Visual Database Systems*, 1995.

[NGC01] H. S. Nagesh, S. Goil, and A. Choudhary. "Adaptive Grids for Clustering Massive Data Sets". In *Proc SIAM Int. Conf. on Data Mining (SDM'01), Chicago, IL*, 2001.

[NH94] R. Ng and J. Han. "Efficient and Effective Clustering Methods for Spatial Data Mining". In *Proc. 20th Int. Conf. on Very Large Databases (VLDB'94), Santiago, Chile*, pages 144–155, 1994.

[NSC04] S. Nassar, J. Sander, and C. Cheng. "Incremental and Effective Data Summerization for Dynamic Hierarchical Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'04), Paris, France)*, 2004.

[OMM+02] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. "Streaming-data Algorithms for High-quality Clustering". In *Proc. 18th Int. Conf. on Data Engineering (ICDE'02), San Jose, CA*, pages 685–704, 2002.

[PJAM02] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. "A Monte Carlo Algorithm for Fast Projective Clustering". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'02), Madison, WI*, 2002.

[PSBK+96] G. Piatetsky-Shapiro, R. Brachman, T. Khabaza, W. Kloesgen, and E. Simoudis. "An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications". In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR*, pages 89–95, 1996.

[PZC+03] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu. "MaPle: A Fast Algorithm for Maximal Pattern-based Clustering". In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM'03), Melbourne, FL*, 2003.

[Sac] Saccharomyces Genome Database (SGD). http://www.yeastgenome.org/. (visited: Oktober/November 2003).

[SEKX98] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications". *Data Mining and Knowledge Discovery*, 2:169–194, 1998.

[Sib73] R. Sibson. "SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method". *The Computer Journal*, 16(1):30–34, 1973.

[SQL+03] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. "Automatic Extraction of Clusters from Hierarchical Clustering Representations". In *Proc. 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2003), Seoul, Korea*, 2003.

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

[THC+99] 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.

[WIY02] D. H. Widyantoro, T. R. Ioerger, and J. Yen. "An Incremental Approach to Building a Cluster Hierarchy". In *Proc. of the 2rd IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan*, pages 705–708, 2002.

[WWYY02] H. Wang, W. Wang, J. Yang, and P. S. Yu. "Clustering by Pattern Similarity in Large Data Set". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'02), Madison, WI*, 2002.

[XEKS98] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases". In *Proc. 14th Int. Conf. on Data Engineering (ICDE'98), Orlando, FL*, pages 324–331, 1998.

[YL01] C. Yang and K.-I. Lin. "An Index Struture for Efficient Reverse Nearest Neighbor Queries". In *Proc. 17th Int. Conf. on Data Engineering (ICDE'01), Heidelberg, Germany*, 2001.

[YWWY02] J. Yang, W. Wang, H. Wang, and P. S. Yu. "Delta-Clusters: Capturing Subspace Correlation in a Large Data Set". In *Proc. 18th Int. Conf. on Data Engineering (ICDE'02), San Jose, CA*, 2002.

[Zah71] C. T. Zahn. "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters". *IEEE Transactions on Computers*, C-20(1), 1971.

[ZRM96] T. Zhang, R. Ramakrishnan, and Livny M. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'96), Montreal, Canada*, pages 103–114, 1996.

[ZS03] J. Zhou and J. Sander. "Data Bubbles for Non-Vector Data: Speeding-up Hierarchical Clustering in Arbitrary Metric Spaces". In *Proc. 29th Int. Conf. on Very Large Databases (VLDB'03), Berlin, Germany*, 2003.

Curriculum Vitae

Peer Kröger was born on January 21, 1975 in Kösching, Germany. He attended primary school from 1981 to 1985, and high-school from 1985 to 1994.

From August 1994 until October 1995, he served in the mandatory civil service at the charitable neighborhood support organization, *Nachbarschaftshilfe* Vaterstetten, Zorneding, and Grasbrunn, Germany.

He entered the *Ludwig-Maximilians-Universität München* (LMU) in November 1995, studying Computer Science with a minor in Physiological Chemistry. His diploma thesis was on "*Molecular Biology Data: Database Overview, Modelling Issues, and Perspectives*", supervised by Prof. Dr. François Bry and Prof. Dr. Rolf Backofen.

In Oktober 2001, Peer Kröger started working at the LMU as a research and teaching assistant in the group of Prof. 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 large standard, spatial, and multimedia databases, data mining for molecular biology data analysis and similarity search in spatial databases.