Coping with new Challenges in Clustering and Biomedical Imaging

Annahita Oswald

München 2011

Coping with new Challenges in Clustering and Biomedical Imaging

Annahita Oswald

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

vorgelegt von
Annahita Oswald
aus München

München, den 23.05.2011

Erstgutachter: Prof. Dr. Christian Böhm

Zweitgutachter: Prof. Dr. Christian Baumgartner

Tag der mündlichen Prüfung: 21.07.2011

Contents

Acknowledgments **ix**

Abstract **xi**

Zusammenfassung **xiii**

I Preliminaries **1**

1 Introduction **3**

1.1 Open Challenges in Clustering and Biomedical Imaging 9

1.1.1 Clustering 9

1.1.2 Biomedical Imaging 11

1.2 Thesis Overview and Contributions 12

2 Survey **15**

2.1 Clustering 16

2.1.1 Partitioning Clustering 17

2.1.2 Hierarchical Clustering 20

2.1.3 Density-based Clustering 23

2.2 Validation of Clustering Results 25

2.3 Classification 27

2.3.1 Support Vector Machines (SVM) 27

2.3.2 K-Nearest Neighbor Classifier (K-NN) 28

2.3.3 Naive Bayes (NB) 28
2.3.4 Decision Tree (DT) 29
2.3.5 Artificial Neutral Networks (ANN) 29
2.3.6 Voting Feature Intervals 30
2.4 Validation of Classification Results 30
2.5 Medical Imaging Technologies 32

II Clustering Techniques 35

3 Hierarchical Clustering 37
3.1 Related Work 39
3.2 Information-Theoretic Cluster Hierarchies 42
3.2.1 Information-theoretic Hierarchical Clustering 43
3.2.2 Hierarchical Cluster Structure 44
3.2.3 Generalization of the MDL Principle 47
3.2.4 Algorithm ITCH 55
3.2.5 Experiments 58
3.3 Genetic Algorithm for Finding Cluster Hierarchies 69
3.3.1 Using Genetic Algorithm for Finding Cluster Hierarchies 71
3.3.2 Algorithm GACH 77
3.3.3 Experiments 77

4 Clustering Mixed Type Data 87
4.1 Introduction 88
4.2 Related Work 89
4.3 Minimum Description Length for Integrative Clustering 92
4.4 Algorithm INTEGRATE 97
4.5 Experiments 100
4.5.1 Synthetic Data 101
4.5.2 Real Data 104
4.5.3 Finding the Optimal k. 105

5 Clustering Skylines 107
5.1 Introduction . . . 108
5.2 Related Work . . . 110
5.2.1 Skyline Computation . . . 110
5.2.2 Sweep-Line Methods . . . 110
5.2.3 Clustering . . . 111
5.3 Theoretical Background . . . 112
5.4 Algorithms to Compute SkyDist . . . 114
5.4.1 SkyDist by Monte-Carlo Sampling . . . 114
5.4.2 SkyDist for 2-Dimensional Skylines . . . 115
5.4.3 A Sweep-Plane Approach for the High-dimensional Case 117
5.5 Experiments . . . 119
5.5.1 Efficiency . . . 119
5.5.2 Clustering Skylines of Real World Data. . . . 120

III Techniques for Mining Biomedical Data 125

6 Detection of Brain Atrophy Patterns based on MRI 127
6.1 Introduction . . . 128
6.2 The FCC framework . . . 129
6.2.1 Feature Selection . . . 130
6.2.2 Clustering . . . 131
6.2.3 Classification . . . 132
6.2.4 Visualization . . . 133
6.3 Experiments . . . 133
6.3.1 Subjects . . . 134
6.3.2 MRI Acquisition . . . 135
6.3.3 MRI Processing . . . 135
6.3.4 Results . . . 137
6.4 Discussion . . . 146

7 Efficient Knowledge Extraction from MRI 153
7.1 Introduction . . . 154
7.2 JGrid/FCC . . . 156
7.2.1 Architecture . . . 157
7.2.2 The FCC Framework . . . 160
7.3 Experiments . . . 162

8 Motif Discovery in Brain Networks 167
8.1 Introduction . . . 168
8.2 Related work . . . 170
8.2.1 Graph Data Set Mining . . . 170
8.2.2 Large Graph Mining . . . 171
8.3 Basic Definitions . . . 173
8.4 Finding Motifs in a Brain Network . . . 174
8.4.1 Construction of Brain Co-Activation Networks Out of fMRI Time Series . . . 175
8.4.2 Performing Frequent Subgraph Mining on Brain Co-Activation Networks . . . 176
8.4.3 Evaluation of Detected Motifs . . . 176
8.5 Experiments . . . 177

IV Conclusions 185

9 Summary and Future Directions 187
9.1 Summary of Contributions . . . 187
9.2 Potentials for Future Work . . . 191

Acknowledgments

I would like to acknowledge all the people who supported me during the development of this thesis. I can only mention some of them here, but my thanks go to all.

First, I would like to thank my supervisor and first referee, Professor Dr. Christian Böhm. I benefited a lot from his enthusiasm for data mining, and enjoyed the inspiring working atmosphere he created. I want to extend my warmest thanks to Professor Dr. Christian Baumgartner for his willingness to be the second referee of this thesis. I would also like to thank the other two members of my thesis committee, Professor Dr. Claudia Linnhoff-Popien and Professor Dr. Hans Jürgen Ohlbach.

My research has profited a lot from the productive discussions and exchange of ideas with my colleagues at the database research group. In particular, I want to thank: Jing Feng, Frank Fiedler, Katrin Haegler, Dr. Tong He, Xiao He, Bettina Konte, Son Mai Thai, Nikola Müller, Dr. Claudia Plant, Michael Plavinski, Junming Shao, Bianca Wackersreuther, Peter Wackersreuther, Qinli Yang and Andrew Zherdin.

This work would never have been possible without Bianca Wackersreuther, who was not only a colleague, coauthor or collaborator, but became a lovely friend. Thank you, for your unselfish help and support!

I do not want to miss all the students I have supervised during my thesis, who supported my work, and who have been beneficial for this thesis. I want to mention here, Timo Becker, Michael Dorn, Sebastian Goebl, Johannes

Huber, Marcel v. Maltitz, Michael Plavinski, Christian Richter, and Felix Sappelt.

I am also grateful to Susanne Grienberger and Franz Krojer for their organizational and technical support during my time at the LMU.

During my thesis, I had the pleasure to work in collaboration with Dr. Michael Ewers and Prof. Dr. Stefan Teipel. This fruitful cooperation had an great impact on some solutions contained in this thesis.

I was honored to be a mentee of the LMU Mentoring program. Professor Dr. Francesca Biagini always supported me and my work and afforded the participation to various conferences.

Lastly, I want to thank my family Kian, Ariane, Inge, and Ebi, as well as my friends for their love and encouragement, advice and support during the last years, and most of all, Ivo, my better half, who always makes me smile.

Annahita Oswald
Munich, May 2011

Abstract

The last years have seen a tremendous increase of data acquisition in different scientific fields such as molecular biology, bioinformatics or biomedicine. Therefore, novel methods are needed for automatic data processing and analysis of this large amount of data. "Data mining" is the process of applying methods like clustering or classification to large databases in order to uncover hidden patterns. Clustering is the task of partitioning points of a data set into distinct groups in order to minimize the intra cluster similarity and to maximize the inter cluster similarity. In contrast to unsupervised learning like clustering, the classification problem is known as supervised learning that aims at the prediction of group membership of data objects on the basis of rules learned from a training set where the group membership is known.

Specialized methods have been proposed for hierarchical and partitioning clustering. However, these methods suffer from several drawbacks. In the first part of this work, new clustering methods are proposed that cope with problems from conventional clustering algorithms. ITCH (Information-Theoretic Cluster Hierarchies) is a hierarchical clustering method that is based on a hierarchical variant of the Minimum Description Length (MDL) principle which finds hierarchies of clusters without requiring input parameters. As ITCH may converge only to a local optimum we propose GACH (Genetic Algorithm for Finding Cluster Hierarchies) that combines the benefits from genetic algorithms with information-theory. In this way the search space is explored more effectively. Furthermore, we propose INTEGRATE

a novel clustering method for data with mixed numerical and categorical attributes. Supported by the MDL principle our method integrates the information provided by heterogeneous numerical and categorical attributes and thus naturally balances the influence of both sources of information. A competitive evaluation illustrates that INTEGRATE is more effective than existing clustering methods for mixed type data. Besides clustering methods for single data objects we provide a solution for clustering different data sets that are represented by their skylines. The skyline operator is a well-established database primitive for finding database objects which minimize two or more attributes with an unknown weighting between these attributes. In this thesis, we define a similarity measure, called SkyDist, for comparing skylines of different data sets that can directly be integrated into different data mining tasks such as clustering or classification. The experiments show that SkyDist in combination with different clustering algorithms can give useful insights into many applications.

In the second part, we focus on the analysis of high resolution magnetic resonance images (MRI) that are clinically relevant and may allow for an early detection and diagnosis of several diseases. In particular, we propose a framework for the classification of Alzheimer's disease in MR images combining the data mining steps of feature selection, clustering and classification. As a result, a set of highly selective features discriminating patients with Alzheimer and healthy people has been identified. However, the analysis of the high dimensional MR images is extremely time-consuming. Therefore we developed JGrid, a scalable distributed computing solution designed to allow for a large scale analysis of MRI and thus an optimized prediction of diagnosis. In another study we apply efficient algorithms for motif discovery to task-fMRI scans in order to identify patterns in the brain that are characteristic for patients with somatoform pain disorder. We find groups of brain compartments that occur frequently within the brain networks and discriminate well among healthy and diseased people.

Zusammenfassung

Die in den letzten Jahren aufgrund neuer Technologien in der Datenerhebung entstandenen Datenmengen gerade in den Bereichen Biologie, Biomedizin oder Bioinformatik, können manuell nicht mehr verarbeitet werden. Deshalb sind Techniken notwendig, die eine automatisierte Auswertung und Analyse der Daten ermöglichen. Im "Data Mining" werden Methoden wie die Clusteranalyse oder die Klassifikation eingesetzt, um bestimmte Muster in einer großen Datenmenge zu finden. Beim Clustering werden Objekte eines Datensatzes in Gruppen (sog. Cluster) eingeteilt, so dass die Ähnlichkeit von Objekten innerhalb eines Clusters minimiert und die Ähnlichkeit zwischen Objekten unterschiedlicher Cluster maximiert wird. Im Gegensatz zur Clusteranalyse (dem nicht überwachten Lernen), werden bei der Klassifikation die Objekte mittels Regeln eingeteilt, die zuvor anhand bereits bekannter und schon klassifizierter Fälle gelernt wurden.

In den letzten Jahren wurden viele Methoden im Bereich des hierarchischen oder partitionieren Clustering entwickelt. Diese Verfahren unterliegen jedoch einigen Schwächen. Im ersten Teil dieser Arbeit sollen daher Clusteringmethoden entwickelt werden, die die Probleme bestehender Verfahren angehen. ITCH (Information-Theoretic Cluster Hierarchies) ist ein hierarchischer Clusteringalgorithmus, der basierend auf einer hierarchischen Variante des Minimum Description Length Prinzips (MDL) eine Clusterhierarchie generiert, wobei keine Eingabeparameter zuvor spezifiziert werden müssen. GACH (Genetic Algorithm for Finding Cluster Hierarchies) verbessert ITCH

dahingehend, dass der Algorithmus auf der Suche nach der optimalen Clusterstruktur nicht in einem lokalen Minimum stecken bleibt. Dies wird durch eine Kombination aus genetischem Algorithmus und Informationstheorie erzielt. Des Weiteren stellen wir INTEGRATE vor, eine neue Clusteringmethode für Daten mit gemischten numerischen und kategorischen Attributwerten. Basierend auf einer informations-theoretischen Optimierungsfunktion für heterogene Attribute, werden die Informationen für numerische und kategorische Attribute in geeigneter Weise einbezogen. Vergleichende Experimente mit anderen Verfahren zeigen eine deutliche Überlegenheit von INTEGRATE auf Datensätzen mit gemischten Datentypen. Mit SkyDist haben wir ein Ähnlichkeitsmaß entwickelt, dass es erlaubt nicht einzelne Objekte miteinander zu vergleichen, sondern ganze Datensätze, die durch ihre Skyline approximiert wurden. Die Sykline eines Datensatzes beschreibt die Menge aller Punkte, die hinsichtlich zwei oder mehrerer Attribute optimal sind. Kombiniert mit Techniken des Data Mining wie Clustering oder Klassifikation, liefert SkyDist gute Einblicke in vielen Anwendungsszenarien.

Der zweite Teil der Arbeit befasst sich mit der Analyse von biomedizinischen Bilddaten, genauer gesagt mit hochauflösenden Magnetresonanz Tomographien (MRT). Dazu haben wir ein System entwickelt, das bestehend aus den Schritten Merkmalsextraktion, Clustering und Klassifikation Muster im Gehirn erkennt, die Kranke von Gesunden unterscheiden. Da allerdings die Verarbeitung solch hochauflösender Tomographien sehr zeitaufwendig ist, haben wir mit JGrid eine Lösung für verteiltes Rechnen vorgestellt, die eine groß angelegte Studie und daher eine verbesserte Diagnosevorhersage ermöglicht. Des Weiteren haben wir in einer anderen Studie ein Verfahren, das häufige Muster innerhalb eines Netzwerks erkennt, auf funktionale MR Bilder von Gesunden und Patienten mit somatoformen Schmerzstörungen angewandt. Wir erhielten Muster im Gehirn, die eine Unterscheidung von Kranken und Gesunden erlauben.

Part I

Preliminaries

Chapter 1

Introduction

With the advent of high-throughput experimental technologies in many scientific domains like biology, medicine, economy, etc. large volumes of data have been captured over the last decade. Large scale analysis of this vast amount of data is not possible manually. In order to automatically "mine" large volumes of data the discipline *Knowledge Discovery in Databases* (KDD) has emerged as a concept in the field of computer science. The core part of the KDD process is called *Data Mining* that aims at finding correlations or patterns among dozens of fields in large databases.

Even in the field of biomedicine, biology, or bioinformatics data mining techniques have been widely applied to understand the mechanisms in the human body that are responsible for diseases like cancer or diabetes prevalent in modern society. The gained information can be used to improve the diagnosis, prevention and treatment of the diseases. In economy, data mining is an essential tool for enhancing productivity, reducing risk and maximizing returns. It is widely believed that data mining will have profound impact on our society and has therefore led to an explosion in demand for novel data mining technologies.

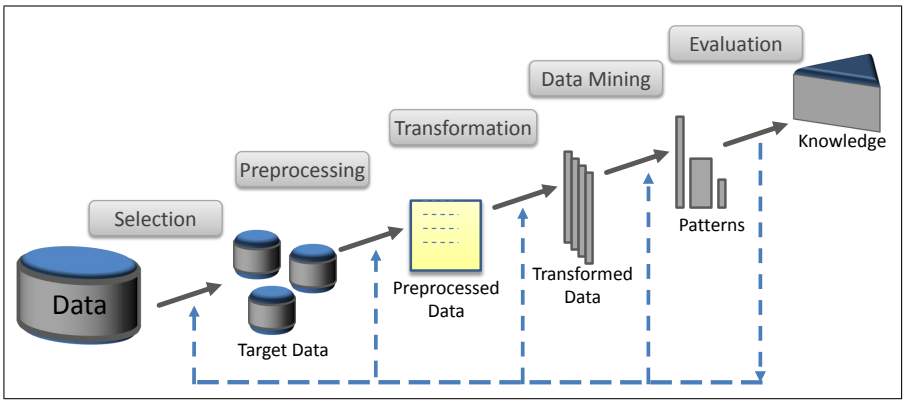


Figure 1.1: The KDD process.

Knowledge Discovery in Databases (KDD) *is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.* [60].

Figure 1.1 illustrates the KDD process which is an iterative sequence of the following steps:

1. **Data Selection.** As a first step, the target data set needs to be created by selecting a data set, or focusing on a subset of several attributes, or data samples.

2. **Preprocessing.** As the target data often is contaminated by noise or unnecessary information the data set needs to be preprocessed before usage. Therefore, data cleaning operations such as removal of noise, normalization of the data or handling missing values have to be applied.

3. **Transformation.** As in most cases only a small subset of the attributes are relevant, techniques for feature selection need to be applied. Feature selection techniques identify those features which are relevant for the goal of the discovery task.

4. **Data Mining.** This is the crucial step of the whole process. Depending on the goal of the KDD process, a suitable data mining method has to be selected and applied to the transformed data.

5. **Evaluation.** The results of the data mining algorithms need to be interpreted and evaluated using expert knowledge or evaluation measures applicable for the respective method. If the result is not satisfactory, there may be the need to return to one of the previous steps, which leads to an iterative process. At the end, the discovered knowledge needs to be documented and reported to interested communities.

Data mining is a core step in the KDD process and therefore often used as a synonym for KDD. In [60], data mining is formulated 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.*

Following Han *et al.* [76] the data mining methods can be categorized as follows:

- **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 appears to be inconsistent with other members of the sample in which it occurs.
- **Classification/Prediction:** prediction of group membership of data objects on the basis of rules learned from a training set where the group membership is known.
- **Association Analysis:** discover relationships between attributes 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:** summarize general features of objects in a data set (characterization), or in a subset of the database and compare particular subsets of the data with comparative subsets (discrimination).

In this thesis, we focus on clustering and classification. Clustering algorithms are essential for knowledge extraction as clusters represent novel knowledge derived from the given database. Clustering can be categorized as an unsupervised data mining task where no class information is required. The objects in the data set are grouped or "clustered" according to its intrinsic structure.

In recent years, many clustering methods have been developed. The most well-known algorithms are described in detail in Section 2.1. The clustering methods can be divided into different categories. Partitioning methods, like k-means [111] or EM [48], generally result in a set of k clusters, each object belonging to one cluster where each cluster may be represented by a summary description of all the objects contained in this cluster. The precise form of this description will depend on the type of the object which is being clustered. In case where real-valued data is available, the arithmetic mean of the attribute vectors for all objects within a cluster provides an appropriate representative; alternative types of centroid may be required in other cases, e.g., a cluster of documents can be represented by a list of those keywords that occur in some minimum number of documents within a cluster.

Hierarchical clustering algorithms find successive clusters using previously established clusters. These algorithms usually are either agglomerative ("bottom-up") or divisive ("top-down"). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

Density-based clustering algorithms are designed to find clusters of arbitrary shape. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). DBSCAN [55] and OPTICS [4] are two typical algorithms of this kind.

In recent years, clustering methods for subspace clustering or projected clustering have been emerged. These methods detect clusters as groups of objects showing high similarity only on a subset of attributes.

In contrast to unsupervised learning, supervised data mining, like classification, aims at predicting class membership of unlabeled objects to predefined classes on the basis of learned models from a training set, where the class membership is known. More precisely, the classifier learns a model on a set of training instances where the class labels are known. This model is then applied to a set of unlabeled objects in order to predict a class membership. The labeling of the training data is often done by experts.

Many classification algorithms have been established in the last years. The most well-known classifiers are the support vector machine, the Naive Bayes Classifier or the Decision Tree classification. A more detailed description on these classifiers is given in Section 2.3.

During last several decades, classification algorithms are applied in various fields. One of the most important tasks is document classification, which group documents into different topics or types according to their similarities of content. One classical example in this line is to distinguish spam messages from legitimate emails. The similarity between documents is usually measured with the associative coefficients from the vector space model, e.g., the cosine coefficient.

In biology or bioinformatics, classification methods are often applied to categorize organisms by biological type, or to assess the taxonomic content of a sample.

In the medical field, classification is mainly applied in medical imaging. Here, different imaging modalities like Radiography, Magnetic resonance imaging (MRI) or Diffusion tensor imaging (DTI) are used to obtain images from the human body. Classifiers can be applied to a set of images in order to make assumptions about the clinical condition of a subject where the diagnosis is not yet known.

Usually, the medical data is high-dimensional, which may comprise millions of features. However, as only a small subset is relevant for the classification task, dimensionality reduction or feature selection techniques have to be applied in order to train a classifier. Selecting the right set of features for classification is a difficult and important problem when designing a good classifier. Typically, one does not know a priori which features are relevant for a particular classification task, and different classifiers may require different feature sets to construct a model for the same problem. The task of finding an optimal subset of features is inherently combinatory. Therefore, feature selection becomes an optimization problem that requires an optimal approach to examine all possible subsets. However, investigating all possible subsets of features is very time consuming even for high-dimensional data. Therefore, computationally simple filter techniques like i.e. the Information Gain, or the χ^2-Statistics are good choices for this kind of data.

In this thesis, we have developed new clustering methods that try to overcome limitations of existing methods. Furthermore, we demonstrate the practical application of data mining methods in the field of medical imaging. Here, we propose the integration of different methods from the field of graph mining, clustering and classification in order to discriminate diseased from healthy subjects.

In the following Section, we elaborate on open challenges in clustering and biomedical imaging, and then provide a broad overview of the main contributions of this thesis in Section 1.2.

1.1 Open Challenges in Clustering and Biomedical Imaging

There are several open challenges in the area of clustering and biomedical imaging, which will be described in this Section.

1.1.1 Clustering

Challenge 1. Parameter-Free Hierarchical Clustering.

Most clustering methods suffer from the drawback that the user has to specify input parameters that usually differ with different data sets. To avoid difficult parameter settings has attracted increased interest in the clustering community in the last years, e.g. [18, 19, 85, 133] just to name a few. Most methods for parameter-free clustering focus on model selection criteria like Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), or the Minimum Description Length (MDL) [73] principle. The idea behind AIC, BIC, and MDL is to penalize model complexity, in addition to deviations from the cluster centers. However, most methods for parameter-free clustering do not provide any cluster hierarchy. The aim is to develop a hierarchical clustering methods that, based on an information-theoretic criterion for model selection, automatically finds an optimal clustering without user-defined parameters.

Challenge 2. Outlier Robustness.

Noise objects are those objects in the data set that do not follow the distribution as the other member of a cluster. On the other hand, outliers are objects in a data set that deviate markedly from other members of the data set. In the presence of outliers or noise objects, many clustering methods fail to detect the true cluster structure that is present in the data. Therefore, outliers should be separated from the cluster objects. Outlier detection is a

challenging task by its own. Especially, measuring the degree of outlierness of single objects has attracted increased interest in the data mining community.

Challenge 3. Sensitivity to Initialization and Local Optima.

The result of most clustering methods depend heavily on the given starting condition or initial state. A common approach is to run the algorithm several times and and rate the quality of the result. Furthermore, the result may only constitute a local optimum. This problem can be solved by exhaustively searching the solution space for the overall optimum result, which leads to high computation time and is not feasible for larger data sets. Genetic algorithms (GA) are based on a stochastic optimization process that thoroughly explores the data space for an optimal result.

Challenge 4. Mixed Type Attributes.

Many objects in real life application scenarios are described by numerous attributes or features. These attributes can be real valued or categorical. The degree in which categorical or numerical information should be used for the clustering task is not trivial. In many existing methods for integrative clustering, this weighting of heterogeneous numerical and categorical information is often done by the user in form of parameters. It is therefore challenging to develop a clustering method that is fully automatic and naturally balances both sources of information provided by numerical and categorical attributes in order to find a good clustering of the heterogeneous data.

Challenge 5. Mining More Complex Data Objects.

The skyline is a well established database primitive for finding database objects which minimize two or more attributes and are therefore the most interesting objects of the data set in some sense. The skyline is highly expressive for many applications. However, most studies focus on efficient algorithms for finding the skyline of a data set and do not use skylines themselves

as objects for data exploration or data mining. Using skylines as objects in data mining tasks such as clustering or classification can give useful insight in many scientific domains. However, no similarity measure for skylines exists so far that can be used in combination with a clustering or classification method.

1.1.2 Biomedical Imaging

Challenge 1. Automated Classification of Alzheimer's Disease in Magnetic Resonance Images.

Demographic changes led to an increasing prevalence of Alzheimer's Disease (AD), the most frequent form of age-related dementia. Mild Cognitive Impairment (MCI) is often regarded as an early stage of AD. The diagnosis of AD or MCI mainly relies on clinical criteria so far. Magnetic Resonance Imaging (MRI) allows to display brain structures with highest resolution and is therefore an important technology in neuroradiology or neuroscience. Sensitive and specific early stage diagnosis of AD is of prime importance to therapeutic interventions. Therefore, data mining and pattern recognition methods are required to extract from millions of voxels of an MRI image the minimal set of voxels that shows systematic abnormalities in subjects with AD or MCI.

Challenge 2. Computational Complexity of Knowledge Extraction.

Most studies for MRI analysis address the quality of the results and the predictive power. However, to get a deeper insight into complex neurological abnormalities like dementia or somatoform pain disorder large-scale analysis is indispensable. Recently, many studies have been proposed that use data from different centers in order to make detailed assumption. These data sets comprise several hundred high-resolution images that have to be processed and analyzed. Therefore, efficient methods are needed in order to manage an efficient processing and analysis of this vast amount of data.

Challenge 3. Common Patterns in Brain Co-Activation Networks.
Idiopathic chronic pain disorders constitute a large, clinically important health care problem that urgently needs deeper pathophysiological insight. The understanding which brain compartments are involved, is a very interesting topic in neurological medicine. Mostly, not only single parts of the brain are the crucial factor in neurological disorders like dementia, schizophrenia or somatoform pain disorder. The challenge is to find those subunits in the brain that interact with each other and govern these neuronal processes.

1.2 Thesis Overview and Contributions

In this thesis, novel clustering methods that can cope with the challenges described in the previous Section are developed, and application of data mining techniques in biomedical data is proposed. In this Section, we provide an overview of the major contributions of this thesis.

Chapter 2 starts with basic notations for clustering and classification and then presents some fundamental algorithms in each field. Furthermore, validation techniques for the evaluation of clustering and classification results are surveyed.

Part II focuses on new challenges in the field of clustering. It presents innovative clustering methods in the field of hierarchical and integrative clustering as well as clustering of more complex objects.

Chapter 3 presents innovative methods for hierarchical clustering based on an information-theoretic criterion for model selection. First, problems of existing hierarchical clustering methods are discussed. Based on these formulations, ITCH (Information-Theoretic Cluster Hierarchies) is presented,

a hierarchical parameter-free clustering algorithm that overcomes these limitations. As ITCH uses a Greedy-like traversal of the hierarchical cluster structure in order to find the optimal one, GACH is proposed, that is based on a genetic algorithm and therefore explores the data space more precisely.

Chapter 4 is dedicated to clustering methods that can cope with mixed numerical and categorical attributes. Therefore, a new information-theoretic criterion for mixed type data is introduced. This can be used as an objective function for integrative clustering. Based on this new quality criterion we present INTEGRATE, a clustering method for data with both numerical and categorical attributes.

Chapter 5 deals with skylines. Traditionally, a single skyline of a data set is determined. We focus on applications where the skylines of different data sets have to be compared. Therefore, a similarity measure for skyline objects is defined which is then integrated into several clustering methods. The experiments demonstrate that using skylines themselves as objects for data mining can give useful insights into many application domains.

Part III is dedicated to mining biomedical data. It describes the application of clustering and classification methods on images from patients with different symptoms to find patterns discriminating healthy from diseased people.

Chapter 6 introduces a framework combining elements from feature selection, clustering and classification (FCC) in order to reveal subtle differences in the brain structure caused by disorders such as Mild Cognitive Impairment, early stage Alzheimer's disease and healthy controls. The results on magnetic resonance images on patient data show excellent accuracies and indicate that FCC is a valuable complement to existing methods.

Chapter 7 proposes the tool JGrid, a highly efficient distributed computing system that allows a distributed computation of the FCC framework on a cluster of computers. In particular, some details about the implementation of the software are presented, including an application to a set of high resolution magnetic resonance images. The experiments demonstrate an tremendous increase in efficiency compared to non-parallelized analysis.

Chapter 8 describes the application of several graph mining algorithms for motif discovery to functional magnetic resonance images of patients with somatoform pain disorder and healthy controls in order to find frequent subgraphs in the brain that are characteristic for the disease. The found motifs represent groups of brain compartment that covary in their activity and differ between diseased and healthy subjects.

Part IV summarizes the contributions of the thesis and points out some directions for future research in this field.

Chapter 2

Survey

The Chapter starts with a general introduction to clustering. In the following Sections, a brief survey on some examples of common partitioning (cf. Section 2.1.1) and hierarchical clustering algorithms (cf. Section 2.1.2), as well as density-based clustering methods (cf. Section 2.1.3) is provided. In Section 2.2, quality measures for comparing different clustering results are presented.

The second part of this Chapter introduces the problem of classification in Section 2.3 and covers some selected classification algorithms in more detail. Section 2.4 deals with the validation of the classification results and with quality measures for comparing classifiers.

As there is a huge variety of clustering and classification algorithms, only methods which are fundamental for the techniques described in this thesis can be surveyed here.

In the last part of this Chapter, we elaborate on some common imaging modalities in the medical field, which are relevant for this thesis (cf. Section 2.5).

2.1 Clustering

Clustering aims at finding groups of objects in a data set, so called clusters, that the intra cluster similarity is minimized while maximizing the inter cluster similarity. Clustering has become very popular in many scientific domains, as it extracts novel knowledge out of a given database.

An important step in clustering is to select a distance measure, which determines how the similarity of two elements is calculated. This influences the shape of the clusters, as some elements may be close to one another according to one distance and further away according to another.
Some common distance functions are the Euclidean, Manhattan, Mahalanobis or Hamming distance.

Given two points x, and y in a data set DS that are described by a d-dimensional feature vector donated by $x = \{x_1, \cdots, x_d\}$ and $y = \{y_1, \cdots, y_d\}$. The distance of these two objects $\vec{x}$ and $\vec{y}$ can be an arbitrary distance function $dist$. Let $dist$ be one of the L_p norms the distance would be defined as follows for an arbitrary $p \in \mathbb{N}$:

$$dist(\vec{x}, \vec{y}) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

The most widely used distance for numerical objects is the Euclidean distance with $p = 2$. In Chapter 5, we have defined SkyDist, which is a similarity measure for more complex object like skyline objects. In all other Chapters, we have used the Euclidean distance for real valued objects.

Clustering methods can be grouped into partitioning, hierarchical and density-based methods. In the following Sections, we present common algorithms of each type.

2.1.1 Partitioning Clustering

In partitioning clustering objects of a data set are grouped into a set of k clusters, each object belonging to one cluster. This Section briefly surveys prominent representatives of partitioning clustering methods.

K-Means

The k-means algorithm was first introduced by James MacQueen in 1967 [111]. It partitions the objects of a data set into k clusters where the number of clusters k has to be predefined by the user.

First the k-means algorithm is initialized by randomly selecting k cluster centers and the points are associated to the nearest cluster center. This point to cluster assignment is then adjusted in an iterative way. In each iteration, the steps recalculating the cluster centers and assigning the points to the nearest center are performed. The cluster centers represent the mean value of the coordinates of all points contained in this cluster. The algorithm stops, if the objective function, squared error function is minimized. The sum of squared distances is a measure for the compactness of a clustering and is defined as follows:

$$\sum_{i=1}^{k} \sum_{j=1}^{n} dist(x_j - \mu_i)^2$$

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. As the clustering depends on the compactness measure the clustering result is very sensitive to noise and outliers. Furthermore, k-means implicitly assumes a Gaussian data distribution of the data, and is thus restricted to detect spherically compact clusters. Another drawback of the algorithm is that the result is significantly sensitive to the initial randomly selected cluster centers resulting in different clusterings with different initializations. In addition to this, the number of

clusters have to be determined in advance. A common approach to find the optimal number of clusters is to run the k-means algorithm several times with different number of k and use a quality or error measure for clustering to obtain the right number of clusters.

EM-clustering

The EM-algorithm [48] is a generalization of the k-means algorithm. In k-means, we attempt to find centroids that are good representatives. We can view the set of k centroids as a model that generates the data. Generating a data set in this model consists of first picking a centroid at random and then adding some noise. If the noise is normally distributed, this procedure will result in clusters of spherical shape. Model-based clustering assumes that the data were generated by a model and tries to recover the original model from the data. More precisely, the data has been generated from a finite mixture of k distributions, but that the cluster membership of each data point is not observed. In the EM-algorithm the log-likelihood is used to calculate the model parameters θ:

$$L(\theta) = \log \prod_{i=1}^{n} P(x_i|\theta) = \sum_{i=1}^{n} \log P(x_i|\theta)$$

The goal of model-based clustering is to determine the parameter $\hat{\theta}$ that maximizes the log-likelihood:

$$L(\hat{\theta}) = \max_{\theta} L(\theta) = \max_{\theta} \sum_{i=1}^{n} \log P(x_i|\theta)$$

In the case of k clusters a vector $\vec{\theta} = \{\theta_1, \cdots, \theta_k\}$ has to be optimized:

$$L(\vec{\theta}) = \sum_{i=1}^{n} \log P(x_i|\ \vec{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \log W_j + \log P(x_i|\theta_j)$$

where W_j is the weight of the cluster that represents the number of objects associated to that cluster.

EM can be applied to many different types of probabilistic modeling. In case the probability density function (PDF) which is associated to a cluster C is a multivariate Gaussian in a d-dimensional data space with parameters μ_C and Σ_C (where $\mu_C = (\mu_{C,1}, ..., \mu_{C,d})^{\mathbf{T}}$, called the location parameter, and Σ_C is a $d \times d$ covariance matrix), the definition of $P(x|\theta)$ where $\theta = \{\mu_1, \cdots, \mu_k, \Sigma_1, \cdots, \Sigma_k\}$ is as follows:

$$P(x|\mu_C, \Sigma_C) = \frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma_C|}} \cdot \mathbf{e}^{-\frac{1}{2}(x-\mu_C)^{\mathbf{T}} \cdot \Sigma_C^{-1} \cdot (x-\mu_C)}$$

The EM algorithm is an iterative procedure where in each iteration step the mixture model of the k distributions are optimized until no further significant improvement of the log-likelihood of the data can be achieved. The EM algorithm has similar drawbacks like k-means. As it is initialized by randomly selecting the parameter vector $\vec{\theta}$ the result heavily depends on the initialization. Furthermore, like k-means the EM algorithm can get stuck in a local maximum of the log-likelihood. However, usually a very fast convergence is observed.

K-Medoid

The k-medoid algorithm is a clustering algorithm related to the k-means algorithm. In contrast to k-means the cluster representatives are not calculated means of the points contained in the cluster, but medoids. A medoid is a object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal i.e. it is a most centrally located point in the given data set. The most common realization of k-medoid clustering are the methods PAM or CLARANS. These algorithms perform two main steps:

1. Initialization, where an initial set of k objects are selected as medoids.

2. Evaluation, where they try to minimize an objective function usually based on the sum of the total distance among non-selected objects and their medoids, i.e., the evaluation step tries to minimize

$$\sum_{i=1}^{n} dist(x_i, m_j)$$

where $s_j \in DS$ and $m_j \in M$, where M denotes the set of medoids and $dist(x_i, m_j) < dist(m_l, x_i), \forall m_j, m_l \in M$ and $m_j \neq m_l$. The smaller the sum of distances among the medoids and all the other objects of their clusters, the better the clustering.

k-medoids based algorithms have been shown to be more robust than k-means since they are less sensitive to the existence of outliers and do not present limitations on continuous attribute types. The drawback of the k-medoids based algorithms is that they are very time consuming and therefore cannot be efficiently applied to large data sets. Furthermore, the number of clusters k has to be selected a priori.

2.1.2 Hierarchical Clustering

In hierarchical clustering the data are not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place, which may run from a single cluster containing all objects to n clusters each containing a single object. Hierarchical clustering is subdivided into agglomerative methods, which proceed by series of fusions of the n objects into groups, and divisive methods, which separate n objects successively into finer groupings. Most hierarchical methods belong to the category of agglomerative clustering. Agglomerative procedures have the drawback that an incorrect merging of clusters in an early stage often yields results which are far away from the real cluster structure. Divisive procedures immediately start with interesting cluster arrangements and are therefore more robust. Since usually agglom-

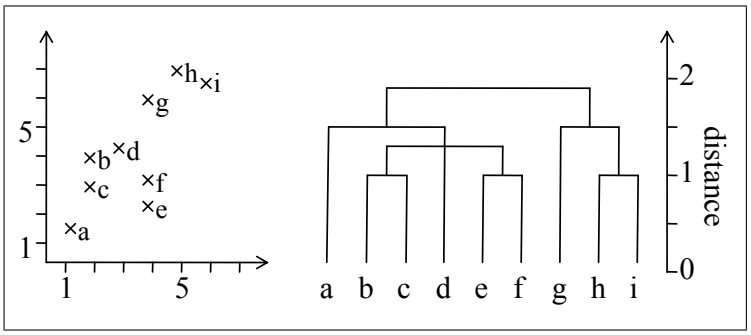


Figure 2.1: Dendrogram (right) for a sample data set (left).

erative procedures are used because of their efficiency, the agglomerative algorithm will be explained in detail in the following.

Single Link

The most well-known agglomerative hierarchical clustering algorithm is the Single Link method [153] which requires no input parameters. Given a data set $DS = \{x_1, \cdots, x_n\}$ the algorithms perform four steps:

1. Place each point $x_i \in DS$ in a single Cluster C_i which results in a set of clusters $C = \{C_1, \cdots, C_n\}$

2. Find the two clusters $C_i, C_j \in C$ with the minimum distance to each other.

3. Merge the clusters C_i and C_j to create a new internal node C_{ij} which will be the parent of C_i and C_j in the resulting dendrogram. Remove C_i and C_j from C.

4. Repeat step 2 and 3 until only one cluster in C is left.

In the first step of the algorithm, when each object represents its own cluster, the distances between the clusters are defined by the chosen distance function, e.g. Euclidean distance. However, once several objects have been

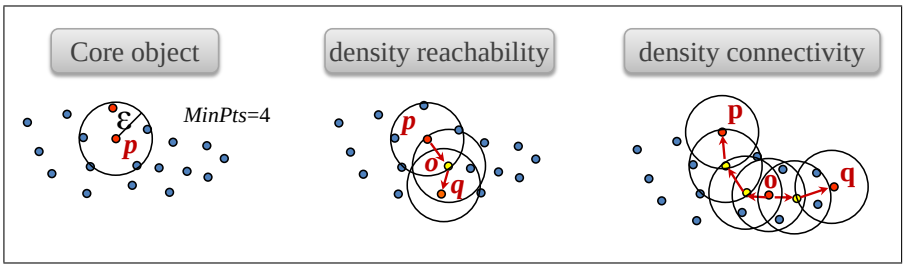


Figure 2.2: Basic notations of DBSCAN.

linked together, a linkage rule is needed to determine the actual distance between two clusters. Single-Link defines the distance $dist_{SL}$ between any two clusters C_i and C_j as the minimum distance between them:

$$dist_{SL}(C_i, C_j) = \min_{x_i \in C_i, y_j \in C_j} dist(x_i, x_j)$$

The hierarchical cluster structure is visualized in a tree like structure the so called dendrogram. The leaves of the dendrogram represent the single data objects, the root the whole data set and the intermediate levels represent the different steps of the cluster merging procedure. Figure 2.1 depicts the dendrogram of an example data set. Using the Single Link method often causes the chaining phenomenon, also called Single Link effect, which is a direct consequence of the Single Link approach tending to force clusters together due to single objects being close to each other regardless of the positions of other entities in that cluster

Alternatives to the Single Link method are the Complete Linkage [44], which defines the maximum distance between two clusters C_i and C_j and the Average Linkage [170], which takes the mean distance between all possible pairs of objects belonging to the two clusters to be merged. However, both alternatives are more time consuming than the Single Link approach.

2.1.3 Density-based Clustering

The key concept of density-based clustering is the observation that inside a cluster the density of points is considerably higher than outside a cluster. Furthermore, different clusters are separated by areas of noise, where the density is lower than inside a cluster. In the following, we present two approaches for density-based clustering, the DBSCAN algorithm which produces a partition of the data set, and OPTICS, a hierarchical extension of DBSCAN.

DBSCAN

The DBSCAN algorithm [55] finds clusters of arbitrary shape and number without requiring the user to specify the number of clusters k. DBSCAN relies on a density-based clustering notion: Clusters are connected dense regions in the feature space that are separated by regions of lower object density. This idea is formalized by two parameters ϵ and $MinPts$, where for each object of a cluster C the neighborhood of a given radius ϵ has to contain at least $MinPts$ objects. These two parameters define the density of the cluster.

Figure 2.2 illustrates the concepts of density-based clustering given the definitions necessary for DBSCAN.

An object is called a *core object* if there exist at least $MinPts$ objects in the ϵ-neighborhood. If one object o is in the ϵ-neighborhood of a core-object p, then o is said to be directly density-reachable from p. The *density-connectivity* is the symmetric, transitive closure of the *direct density reachability*, and a density-based cluster is defined as a maximal set of *density connected* objects.

In contrast to many other partitioning clustering methods such as k-means and k-medoid methods, DBSCAN can detect clusters of arbitrary shape and is robust against noise. However, the clustering result strongly depends on an appropriate choice of the parameters ϵ and $MinPts$.

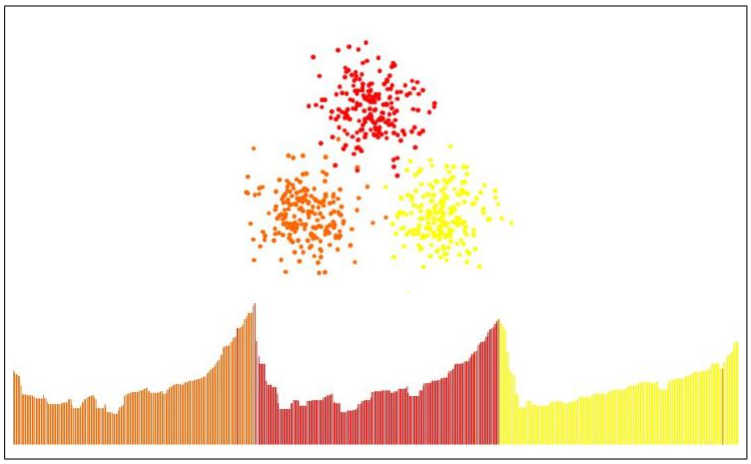


Figure 2.3: Reachability plot of OPTICS for three Gaussian clusters.

OPTICS

OPTICS [4] is the hierarchical extension to DBSCAN and avoids the chaining effect of Single Link by requiring a minimum objects density. Like DBSCAN, OPTICS requires the two parameters: ϵ, which describes the maximum distance (radius) to consider, and $MinPts$, describing the number of points in the ϵ hyper-sphere.

The main idea of OPTICS is, that (for a constant $MinPts$-value) density-based clusters w.r.t. a higher density are completely contained in density-based clusters w.r.t. a lower density. Therefore, 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 during a single traversal of the data set. In contrast to DBSCAN, OPTICS does not assign cluster memberships, but stores linear order of the data objects according their hierarchical cluster structure. This cluster structure can be visualized by a so-called 2-dimensional reachability plot where the objects are plotted according to the sequence specified in the cluster ordering along the x-axis, and for each object, its reachability along the y-axis. Figure 2.3 depicts the

reachability plot based on the cluster ordering computed by OPTICS for the sample 2-dimensional data set that comprises three Gaussian clusters. Valleys in this plot indicate clusters: objects having a small reachability value are closer and thus more similar to their predecessor objects than objects having a higher reachability value.

2.2 Validation of Clustering Results

To compare the performance of different clustering methods usually a class label has to be assigned to each object which is not used for clustering but only validation purposes. Standard cost functions, such as the sum of squared distances from the cluster centers, depend on metric interpretations of the data. They cannot be used to compare techniques that use different distance measures. Information-theoretic measures can be used to indicate the match between cluster labels and class labels. In the following, we present four different measures that we used in our experiments.

Most information-theoretic measures are based on the mutual information (MI) of a clustering. Given two clusterings $C = \{C_1, \cdots, C_u\}$ and $C' = \{C'_1, \cdots, C'_v\}$ of a data set DS and a contingency table $M = [m_{ij}]_{j=1\ldots v}^{i=1\ldots u}$ where m_{ij} denotes the number of objects that are common to clusters C_i and C'_j. Based on this contingency table, various cluster similarity measures can be built. The mutual information is a symmetric measure that quantifies the mutual dependence between two random variables, or the information that C and C' share, and is defined as follows:

$$MI(C, C') = \sum_{i=1}^{u} \sum_{j=1}^{v} p(i,j) log \frac{p(i,j)}{p(i)p(j)}$$

where $p(i,j)$ denotes the probability that a point belongs to cluster $C_i \in C$ and cluster $C'_j \in C'$. It quantifies the information shared by the two clusterings and thus can be used as similarity measure in clustering.

The Normalized Mutual Information (NMI) [158] is a normalized version of the mutual information. The mutual information of two clusterings C and C' is divided by the maximum value of the index. It has fixed upper and lower bounds and takes a value of 1 if two clusterings are identical and 0 if the two clusterings are independent.

The Adjusted Mutual Information (AMI) [127] is based on the Expected Mutual Information (EMI), has fixed bounds, and is corrected for randomness. It takes a value of 1 if the two clusterings are identical and 0 if the mutual information between the two clusterings equals its expected value.

The Expected Mutual Information (EMI) [127]. Given two clustering C and C' the EMI defines the average mutual information between all clustering pairs that have the same number of clusters and objects in each cluster as in C and C' respectively. Thus, the average of mutual information value between all possible pairs of clustering is actually the expected value of $MI(M)$ over the set of associated contingency tables $\mathcal{M}$. To generate two random clusterings is done using the permutation model described in [104]. This ensures to correct the measure for randomness.

DOM *et al.* [49] proposed a measure, referred to as DOM in the following, that corresponds to the number of bits required to encode the class labels when the cluster labels are known. If the number of clusters is equal in both clusterings C and C' the measure is equal to the mutual information. In cases where the number of clusters are different it computes the reduction in number of bits that would be required to encode the class labels. Small values for the value refer to good clusterings. We refer to DOM in the rest of this thesis.

2.3 Classification

Classification aims at learning a mapping from a vector of features $\vec{x}$ to a categorical variable l. The variable to be predicted is called the class label. Therefore, a training data set is used in order to learn a model. This data set used for training is of the form $\{x_1, l_1, \cdots, x_n, l_n\}$, where each training instance $x_i \in DS_{TRAIN}$ is assigned a class label $l_i \in L$. This labeling is often done by experts. The goal is, to learn a function $f : DS_{TRAIN} \rightarrow L$ that maps as much objects $\vec{x}$ of a set of objects DS_{TRAIN} to their correct class $l \in L$. Many classification methods require a metric distance function *dist* as specified in Section 2.1.
In the following, we will describe some common classification algorithms.

2.3.1 Support Vector Machines (SVM)

A Support Vector Machine [38] is a non-probabilistic binary classifier that performs classification by constructing a d-dimensional maximum margin hyperplane that optimally separates the data into two groups. A good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class, since in general the larger the margin the lower the generalization error of the classifier. Whereas the original problem may be stated in a finite dimensional space, it often happens that in that space the sets to be discriminated are not linearly separable. For this reason, it was proposed that the original finite dimensional space be mapped into a much higher dimensional space, presumably making the separation easier in that space. SVM uses a mapping into a larger space so that cross products may be computed easily and the data can be separated linearly. As an alternative to maximum margin hyperplane, a soft margin can be used by specifying a cost factor c which enables the separation of data that can not be separated linearly. A way to create a non-linear classifier is to replace the cross product by a nonlinear kernel function. This allows the algorithm to fit

the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space.

SVM is widely used for classification of high-dimensional data which often occurs in the field of biology or medicine.

2.3.2 K-Nearest Neighbor Classifier (K-NN)

The most basic instance-based method is the k-nearest neighbor method (kNN). The prediction of class membership for each unlabeled object is based on the majority class of the k closest training objects. One refinement to the kNN method is to weight contribution of each of the k neighbors according to their distance giving greater weight to closer neighbors.

kNN is robust against noisy training data and quite effective when it is provided a sufficiently large set of training data. However, no model is learned and thus no information about the training data is given as a result [120].

2.3.3 Naive Bayes (NB)

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong independence assumptions of single features. For classification, an object is assigned a class label based on the probability of each class given the object's feature. However, the independence assumption is not realistic in many applications. An advantage of the Naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Furthermore, as independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix [120, 89].

2.3.4 Decision Tree (DT)

In Decision tree learning, classification is based on a learned function that is represented by a decision tree. Each node in the tree specifies a test of some attribute, and each branch descending from the node corresponds to one of the possible values for this attribute. The leaves specify the class labels. An object is classified by traversing the tree by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute. This process is repeated until the leave node is reached which species the class label. In general, each path from the root to the leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions. To build the tree, information gain is used to select the candidate attributes that best separate the training data according to their class label in each split [145, 146].

2.3.5 Artificial Neutral Networks (ANN)

Artificial neural networks are relatively crude electronic networks of several layers of *neurons* based on the neural structure of the brain. In general, the first layer has input neurons, which send data via synapses to the hidden layers of neurons, and then via more synapses to the third layer of output neurons. The synapses store weights that manipulate the data in the calculations. The standard algorithm used for classification is a multi-layered ANN that is trained using the backpropagation algorithm and the delta rule.

Neural networks are data driven self-adaptive methods in that they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model. Classification with ANN is very efficient, even though the time needed for training is very high. However, too many hidden layers may lead to overfitting [120].

2.3.6 Voting Feature Intervals

Demiröz *et al.* [47] proposed the classification by voting feature intervals (VFI), a simple entropy-based classifier that constructs intervals for each class and each feature and records class counts. Each interval can be represented as a vector $< lower, count_1, \cdots, count_k >$, where $lower$ denotes the lower bound and $count_1, \cdots, count_k$ the votes for each class, thus, the number of objects of each class having an feature value of f_i within the interval. Therefore, each feature participates in the classification by distributing real-valued votes among classes. Classification is performed by voting. The class receiving the highest vote is declared to be the predicted class.

2.4 Validation of Classification Results

To estimate the performance of a classifier two different validation techniques can be applied.

Train and Test. Here, the data set is partitioned into a training and test set. The classifier is trained on the training set and the obtained model is then applied to the test set.

n-Fold Cross-Validation. This validation technique is often used in the case of few training data. Therefore, the training data set is separated into n subsets of equal size. The classifier is trained on $n - 1$ subsets and the remaining subset is used for testing. The cross-validation process is then repeated n times (the folds), with each of the n subsets used exactly once as the validation data. In each fold quality measures are averaged to produce a single estimation.

There are many different quality measures available for evaluating the classification result. We focus on the most common ones that are used in this thesis.

Accuracy denotes the proportion of true results (both true positives and

true negatives) and is given by the following formula:

$$acc = \frac{TP + TN}{|DS|}$$

Precision can be seen as a measure of exactness and is defined as the proportion of the true positives against all the positive results (both true positives and false positives):

$$prec = \frac{TP}{TP + FP}$$

Recall is a measure of completeness. The definition of the recall is defined as the proportion of the true positives against all true positives and false negatives:

$$recall = \frac{TP}{TP + FN}$$

In case of a binary classification the measures Sensitivity and Specificity can be used. In biomedical application, classification often aims at separating diseased instances from a healthy control group. The test outcome can be positive (predicting that the person has the disease) or negative (predicting that the person does not have the disease). The test results for each subject may or may not match the subject's actual state.

- True positive: Diseased subjects correctly diseased as sick
- False positive: Healthy people incorrectly identified as sick
- True negative: Healthy people correctly identified as healthy
- False negative: Diseased people incorrectly identified as healthy.

Sensitivity measures the proportion of actual positives which are correctly identified as such (e.g. the percentage of subjects who are correctly identified as having the disease).

$$sen = \frac{TP}{TP + FN}$$

Specificity measures the proportion of negatives which are correctly identified (e.g. the percentage of healthy subjects who are correctly identified as not having the disease).

$$spec = \frac{TN}{TN + FP}$$

The sensitivity of a classifier, also called the true positive rate corresponds to the recall of the class representing the diseased individuals. Analogously, the specificity or true negative rate corresponds to precision of the healthy class.

2.5 Medical Imaging Technologies

In medicine, different imaging modalities are applied for clinical purposes. In the following, we describe the most common imaging technologies that are used in the clinical context for diagnosis or treatments of diseases.

Computed Tomography (CT)

Computed Tomography is a powerful tool in diagnostic medicine producing three-dimensional cross-sectional images of the inside of an object. The image is generated from a large series of two-dimensional X-ray images taken around a single axis of rotation [79]. CT scanning of the head is typically used to detect infarction, tumors, calcifications, and bone trauma. There are several advantages that CT has over traditional 2-dimensional medical radiography. CT completely eliminates the superimposition of images of structures outside the area of interest. Moreover, even subtle differences between body tissues can be detected because of the inherent high-contrast resolution of CT.

However, CT is regarded as a moderate- to high-radiation diagnostic technique [25] [28].

Magnetic Resonance Imaging (MRI)

In contrast to Computed Tomography, magnetic resonance imaging does not

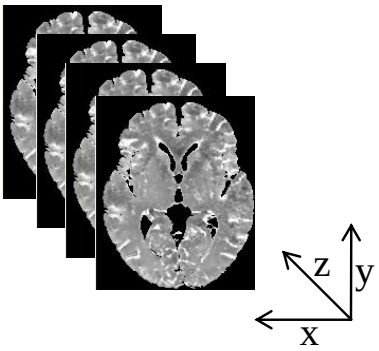


Figure 2.4: Slices of an MRI scan.

expose subjects to radiation. MRI uses strong magnetic fields to align atomic nuclei within body tissues, then uses a radio signal to alter the axis of rotation of these nuclei and observes the radio frequency signal generated as the nuclei return to their baseline states. Like CT, MRI traditionally creates a 2-dimensional image of a thin "slice" of the body. Figure 2.4 exemplarily shows some of the slices of an MR image of the human brain. Each feature in the image is called voxel which is defined by the location in x and y direction and the slice z.

MRI scans provide best contrast between different soft tissues of the body. Thus, MRI is widely used in imaging the brain, muscles, the heart, and cancers.

Functional Magnetic Resonance Imaging (fMRI) is a type of specialized MRI scan that uses MR imaging to measure metabolic changes that take place in an active part of the brain. fMRI cannot detect absolute activity of brain regions. It can only detect difference of brain activity between several conditions. During the fMRI image acquisitions, the patient has to perform several tasks or is stimulated to trigger several processes or emotions. These conditions are repeated several times and can be separated by rest periods [128].

Diffusion Tensor Imaging (DTI)

DTI uses magnetic resonance imaging (MRI) to allow measuring restricted diffusion of water through tissue. DTI is a new but rapidly evolving imaging technology that is able to determine the direction of water diffusion within cellular structures, such as nerve bundles, or fibers. Basically, magnetic field variations of the MRI magnet are applied in at least six different directions, which makes it possible to calculate a tensor for each voxel that shows the three dimensional shape of the diffusion pattern.

DTI data not only yield information on the integrity of cellular structures, but this data can also be used to create mathematical representations of the cellular structures (such as nerve fibers) in three dimensions, which can be used for diagnosis of specific diseases. Therefore, with DTI, white matter lesions can be found that do not show up on other imaging techniques, and can also be used to localize tumors. The most advanced application is certainly that of fiber tracking in the brain which provides exciting new opportunities to study the anatomy of the central nervous system [105].

Part II

Clustering Techniques

Chapter 3

Hierarchical Clustering

Hierarchical clustering methods are widely used in various scientific domains such as molecular biology, medicine, or economy as dendrograms and similar hierarchical representations provide extremely useful insights into the structure of a data set. Therefore, a large number of hierarchical clustering methods such as Single Link [153] and its variants [125] or OPTICS [4] have been proposed. Please refer to Chapter 2.1.2 for a detailed discussion.

A related problem of many existing approaches is that they are order-sensitive as they may generate different clusters for different orders of the same input data. Other methods heavily depend on the initialization or may converge only to a local optimum as they suffer from the inability to perform adjustments once the splitting or merging decision is made. Single Link and OPTICS describe the result in form of a dendrogram or a reachability plot. These hierarchical representations is often hard to interpret even for large data sets. Many methods that overcome the problems need to specify a non-intuitive parameter like the number of clusters or the minimum object density for clustering in OPTICS.

Following four goals are not yet fully satisfied by previous methods: First, to guide the hierarchical clustering algorithm to identify only meaningful and

valid clusters. Second, to represent each cluster in the hierarchy by an intuitive description with, e.g. a probability density function. Third, to consistently handle outliers, ideally by identifying and separating them. Fourth, to avoid difficult parameter settings in order to provide a fully automatic hierarchical clustering method. And finally, to be flexible enough finding the correct hierarchical cluster structure of the data. Most of the existing methods address one or more of these goals, but do not satisfy all of them.

In this Chapter, two new approaches for hierarchical clustering are proposed. First, Section 3.1 surveys related work in the area of hierarchical, model-based, information-theoretic clustering and genetic algorithms. Then, Section 3.2 introduces the algorithm ITCH (Information-Theoretic Cluster Hierarchies). ITCH is a novel clustering method that is built on a hierarchical variant of the information-theoretic principle of Minimum Description Length (MDL), referred to as hMDL. Interpreting the hierarchical cluster structure as a statistical model of the data set, it can be used for the effective data compression by Huffman coding. Using the compression rate as an objective function for clustering enables ITCH to fulfill the first four goals mentioned above.

In Section 3.3 the algorithm GACH (Genetic Algorithm for Finding Cluster Hierarchies) is presented which includes all advantages of ITCH, and furthermore, can cope with the last goal mentioned before. GACH integrates the idea of a Genetic Algorithm (GA) that is a stochastic optimization technique based on the mechanism of natural selection and genetics. Using this GA-based stochastic search GACH thoroughly explores the solution space more effectively than existing methods. GACH does not depend on the initialization and is flexible enough to find the correct hierarchy that is present in the data.

3.1 Related Work

Hierarchical Clustering. One of the most widespread approaches to hierarchical clustering is the Single Link algorithm [153] and its variants [170, 44]. The resulting hierarchy obtained by the merging order is visualized as a tree, which is called dendrogram. Cuts through the dendrogram at various levels obtain partitioning clusterings. However, for complex data sets it is hard to define appropriate splitting levels, which correspond to meaningful clusterings. Furthermore, outliers may cause the well-known Single Link effect. OPTICS [4] avoids the Single Link effect by requiring a minimum object density for clustering, i.e. $MinPts$ number of objects are within a hypersphere with radius ϵ. Additionally, it provides a more suitable visualization, the reachability plot. However, the problem that only certain cuts represent useful clusterings still remains unsolved.

Model-based Hierarchical and Semi-supervised Clustering. [167] proposes a hierarchical extension of the EM algorithm [48] to speed up query processing in an object recognition application. In [29] a hierarchical variant of EM is applied for image segmentation. Goldberger and Roweis [71] focus on reducing the number of clusters in a mixture model. The consistency with the initial clustering is assured by the constraint that objects belonging to the same initial cluster must end up after the reduction in the same new cluster as well. Each of these approaches needs a suitable parameter setting for the number of hierarchy levels. Clustering respecting some kind of hierarchy can also be regarded as *semi-supervised clustering*, i.e. clustering with side information. In most of some recent studies [110, 16, 13], this information is introduced by constraints on the objects and typically consists of strong expert knowledge.

Information Theory in the Field of Clustering. Only a few papers on compression based clustering, that avoid difficult parameter settings have

been published so far. X-means [133], G-means [75], RIC [18] and OCI [19] focus on avoiding the choice of k in partitioning clustering by trying to balance data likelihood and model complexity. This sensitive trade-off can be rated by model selection criteria, among them the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC) and Minimum Description Length (MDL) [73]. X-means provides a parameter-free detection of spherical Gaussian clusters by a top-down splitting algorithm, which integrates k-means clustering and BIC. G-means extends this idea to detect non-spherical Gaussian clusters. The model selection criterion of RIC and OCI is based on MDL, which allows to define a coding scheme for outliers and to identify non-Gaussian clusters.

There is a family of further closely related ideas, such as Model-based Clustering [11], the work of Still and Bialek [156] and the so-called Information Bottleneck Method [165], introduced by Tishby *et al.* This technique aims at providing a quantitative notation of *meaningful* or *relevant* information. The authors formalize this perception by finding the best trade-off between accuracy and complexity when clustering a random variable X, given a joint probability distribution between X and an observed relevant variable Y. It is used for clustering terms and documents [155]. However, all parameter-free algorithms mentioned so far, do not provide any cluster hierarchy. One recent paper [35] presents a new method for clustering based on compression. In the first step, this method determines a parameter-free, universal, similarity distance, computed from the lengths of compressed data files. Afterwards a hierarchical clustering method is applied. However, this work was not designed to handle outliers in an appropriate way. Furthermore, no description of the content of a cluster is available.

Genetic Clustering Algorithms. Krishna and Murty [99] introduced a genetic k-means algorithm (GKA), which determines a global optimal partitioning of the given data into a specified number k of clusters. In their

approach the clusters inside a chromosome are encoded by strings that hold the IDs of the clusters in a given order. The selection function of GKA is the so-called weighted roulette wheel. The total within-cluster variation serves as a fitness indicator. Paul Scheunders [152] published a genetic variant of the c-means clustering algorithm (GCMA). The genetic approach affects the run-time drastically resulting in a trade-off between run time and quality of the result. Some kind of semi-supervised evolutionary clustering were presented by [46], which is also a k-means based approach. In [109] the optimization of a clustering is tried to be achieved by using two fitness functions. Besides that, the building block construction has a major influence on the behavior of the genetic algorithm. Thus, this algorithm tries to improve a fitness function concerning the space restrictions on the one hand and concerning the building blocks on the other hand. One recent approach is the one by Pernkopf and Bouchaffra [136], which combines the benefits of a GA with model-based clustering to find a nearly optimal solution for a given number of clusters. With the help of a MDL criterion the correct number of clusters is determined fully automatically.

All these methods are only applicable to partitioning clustering and suffer from the problem that human interaction is still necessary to enter a suitable k for the number of clusters. The detection of noise and outliers is not supported at all.

3.2 Information-Theoretic Cluster Hierarchies

In recent years many algorithms for finding hierarchical dependencies between clusters in a data set have been proposed. However, these methods often fail to detect the true cluster structure that is present in a data set even in the presence of noise or outliers. "So how can we decide if a given representation is really natural, valid, and therefore meaningful?" and "How can we enforce a hierarchical clustering algorithm to identify only the meaningful cluster structure?" We give the answer to these questions by relating the hierarchical clustering problem to that of information theory and data compression where we interpret the cluster hierarchy as a statistical model of the data set, which defines more or less likely areas of the data space. The knowledge of these probabilities can be used for an efficient compression of the data set: Following the idea of (optimal) Huffman coding, we assign few bits to points in areas of high probability and more bits to areas of low probability. The compression becomes the more effective, the better our statistical model, the hierarchical cluster structure, fits to the data. In this Chapter a new hierarchical clustering algorithm, called ITCH is proposed that relies on a hierarchical variant of the MDL criterion (hMDL) and overcomes the following limitations of existing approaches:

1. All single clusters as well as their hierarchical arrangement are guaranteed to be **meaningful**. Nodes only are placed in the cluster hierarchy if they improve the data compression. This is achieved by optimizing the hMDL criterion. Moreover, a maximal consistency with partitioning clustering methods is obtained.

2. Each cluster is represented by an **intuitive description** of its content in form of a Gaussian probability density function (PDF). The output of conventional methods is often just the (hierarchical) cluster structure and the assignment of points.

3. ITCH is **outlier-robust**. Outliers are handled by assigning them to the

root of the cluster hierarchy or to an appropriate inner node, depending on the degree of outlierness.

4. ITCH is **fully automatic** meaning that no difficult parameter settings are necessary.

The rest of this Chapter is organized as follows. Section 3.2.1 introduces the hMDL criterion underlying ITCH. Section 3.2.4 provides the details of the new ITCH algorithm. Several comparative experiments using synthetic and real-world data sets show the performance and the effectiveness of ITCH in Section 3.2.5. Parts of the material presented in this Chapter have been published in [20].

3.2.1 Information-theoretic Hierarchical Clustering

The clustering problem is highly related to that of data compression: The detected cluster structure can be interpreted as a PDF $f_\Theta(x)$ where $\Theta = \{\theta_1, \theta_2, ...\}$ is a set of parameters, and the PDF can be used for an efficient compression of the data set. It is well-known that the compression by *Huffman coding* is optimal if the data distribution really corresponds to $f_\Theta(x)$. Huffman coding represents every point x by a number of bits which is equal to the negative binary logarithm of the PDF:

$$C_{\text{data}}(x) = -\log_2(f_\Theta(x)).$$

The better the point set corresponds to $f_\Theta(x)$, the smaller the coding costs $C_{\text{data}}(x)$ are. Hence, $C_{\text{data}}(x)$ can be used as the objective function in an optimization algorithm. However, in data compression, Θ serves as a *code book* which is required to decode the compressed data set again. Therefore, we need to complement the compressed data set with a coding of this code book, the parameter set Θ. When, for instance, a Gaussian Mixture Model (GMM) is applied, Θ corresponds to the weights, the mean vectors and the

variances of the single Gaussian functions in the GMM. Considering Θ in the coding costs is also important for the clustering problem, because neglecting it leads to overfitting. For partitioning (non-hierarchical) clustering structures, several approaches have been proposed for the coding of Θ [73, 133]. These approaches differ from each other because there is no unambiguous and natural choice for a distribution function, which can be used for the Huffman coding of Θ, and different assumptions lead to different objective functions. In case of the hierarchical cluster structure in ITCH, a very natural distribution function for Θ exists: With the only exception of the root node, every node in the hierarchy has a parent node. This parent node is associated to a PDF which can naturally be used as a code book for the mean vector (and indirectly also for the variances) of the child node. The coding costs of the root node, however, is not important, because every valid hierarchy has exactly one root node with a constant number of parameters, and therefore, the coding costs of the root node is always constant.

3.2.2 Hierarchical Cluster Structure

In this Section, we formally introduce the notion of a hierarchical cluster structure (HCS). A HCS contains clusters $\{A, B, ...\}$ each of which is represented by a Gaussian distribution function. These clusters are arranged in a tree:

Definition 1 (Hierarchical Cluster Structure) *(1) A HCS is a tree* $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ *consisting of a set of nodes* $\mathcal{N} = \{A, B, ...\}$ *and a set of directed edges* $\mathcal{E} = \{e_1, e_2, ...\}$ *where A is a parent of B (B is a child of A) iff* $(A, B) \in \mathcal{E}$*. Every node* $C \in \mathcal{N}$ *is associated to a weight* W_C *and a Gaussian PDF defined by the parameters* μ_C *and* Σ_C *such that the sum of the weights equals one:*

$$\sum_{C \in \mathcal{N}} W_C = 1.$$

(2) If a path from A to B exists in $\mathcal{T}$ (or $A = B$) we call A an ancestor of B (B a descendant of A) and write $B \sqsubseteq A$.
(3) The level l_C of a node C is the height of the descendant subtree. If C is a leaf, then C has level $l_C = 0$. The root has the highest level (length of the longest path to a leaf).

The PDF which is associated to a cluster C is a multivariate Gaussian in a d-dimensional data space which is defined by the parameters μ_C and Σ_C (where $\mu_C = (\mu_{C,1}, ..., \mu_{C,d})^{\mathbf{T}}$ is a vector from a d-dimensional space, called the location parameter, and Σ_C is a $d \times d$ covariance matrix) by the following formula:

$$N(\mu_C, \Sigma_C, x) = \frac{1}{\sqrt{(2\pi)^d \cdot |\Sigma_C|}} \cdot \mathbf{e}^{-\frac{1}{2}(x-\mu_C)^{\mathbf{T}} \cdot \Sigma_C^{-1} \cdot (x-\mu_C)}.$$

For simplicity we restrict $\Sigma_C = \text{diag}(\sigma_{C,1}^2, ..., \sigma_{C,d}^2)$ to be diagonal such that the multivariate Gaussian can also be expressed by the following product:

$$\begin{aligned} N(\mu_C, \Sigma_C, x) &= \prod_{1 \leq i \leq d} N(\mu_{C,i}, \sigma_{C,i}^2, x_i) \\ &= \prod_{1 \leq i \leq d} \frac{1}{\sqrt{2\pi\sigma_{C,i}^2}} \cdot \mathbf{e}^{-\frac{(x_i - \mu_{C,i})^2}{2\sigma_{C,i}^2}}. \end{aligned}$$

Since we require the sum of all weights in a HCS to be 1, a HCS always defines a function whose integral is ≤ 1. Therefore, the HCS can be interpreted as a complex, multimodal, and multivariate PDF, defined by the mixture of the Gaussians of the HCS $\mathcal{T} = (\mathcal{N}, \mathcal{E})$:

$$f_{\mathcal{T}}(x) = \max_{C \in \mathcal{N}} \{W_C \, N(\mu_C, \Sigma_C, x)\} \text{ with} \int_{\mathbb{R}^d} f_{\mathcal{T}}(x) \mathbf{d}x \leq 1.$$

If the Gaussians of the HCS do not overlap much, then the integral becomes close to 1.

The operations, described in Section 3.2.4, assign each point $x \in DB$ to a cluster of the HCS $\mathcal{T} = (\mathcal{N}, \mathcal{E})$. We distinguish between the *direct* and the *indirect* association. A point is directly associated to that cluster $C \in \mathcal{N}$ the probability density of which is maximal at the position of x, and we write $C = Cl(x)$ and also $x \in C$, with:

$$Cl(x) = \arg\max_{C \in \mathcal{N}} \{W_C \cdot N(\mu_C, \Sigma_C, x)\}.$$

One of the main motivations of our hierarchical, information-theoretic clustering method ITCH is to represent a sequence of clustering structures which range from a very coarse (unimodal) view to the data distribution to a very detailed (multi-modal) one, and that all these views are meaningful and represent an individual complex PDF. The ability to cut a HCS at a given level L is obtained by the following definition:

Definition 2 (Hierarchical Cut) *A HCS $\mathcal{T}' = (\mathcal{N}', \mathcal{E}')$ is a hierarchical cut of a HCS $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ at level L (in symbols: $\mathcal{T}' = HC_L(\mathcal{T})$), if the following properties hold:*
(1) $\mathcal{N}' = \{A \in \mathcal{N} | l_A \geq L\}$,
(2) $\mathcal{E}' = \{(A, B) \in \mathcal{E} | l_A > l_B \geq L\}$,
(3) For each $A \in \mathcal{N}'$ the following properties hold:

$$W'_A = \begin{cases} W_A & \text{if } l_A > L \\ \sum_{B \in \mathcal{N}, B \sqsubseteq A} W_B & \text{otherwise,} \end{cases}$$

where W_C and W'_C is the weight of node C in $\mathcal{T}$ and $\mathcal{T}'$, respectively.
(4) Analogously, for the direct association of points to clusters the following property holds: Let x be associated to Cluster B in $\mathcal{T}$, i.e. $Cl(x) = B$. Then in $\mathcal{T}'$, x is associated to:

$$Cl'(x) = \begin{cases} B & \text{if } l_B \geq L \\ A | B \sqsubseteq A \wedge l_A = L & \text{otherwise.} \end{cases}$$

Here, the weights of the pruned nodes are automatically added to the leaf nodes of the new HCS, which used to be the ancestors of the pruned nodes. Therefore, the sum of all weights is maintained (and still equals 1), and the obtained tree is again a HCS according to Definition 1. The same holds for the point-to-cluster assignments.

3.2.3 Generalization of the MDL Principle

So how can we generalize the MDL principle for hierarchical clustering? Our new objective function hMDL ensures that in the HCS nodes only pay off if they optimize the overall data compression. Following the traditional MDL principle, we compress the data points according to their negative log likelihood corresponding to the PDF which is given by the HCS. In addition, we penalize the model complexity by adding the code length of the HCS parameters to the negative log likelihood of the data. In a HCS, we can exploit the fact that the parameters of a child node typically fit well into the PDF of the parent node. The better the PDFs of child nodes fit into the PDFs of the parent, the less the coding costs will be. Therefore, the overall coding costs corresponds to the natural notion of a good hierarchical representation of data by distribution functions. The discrete assignment of points to clusters allows us to determine the coding costs of the points clusterwise and dimensionwise, as explained in the following: The coding costs of the points associated to the clusters $C \in \mathcal{N}$ of the HCS $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ corresponds to:

$$C_{\text{data}} = -\log_2 \prod_{x \in DB} \max_{C \in \mathcal{N}} \left\{ W_C \prod_{1 \leq j \leq d} N(\mu_{C,j}, \sigma^2_{C,j}, x_j) \right\}.$$

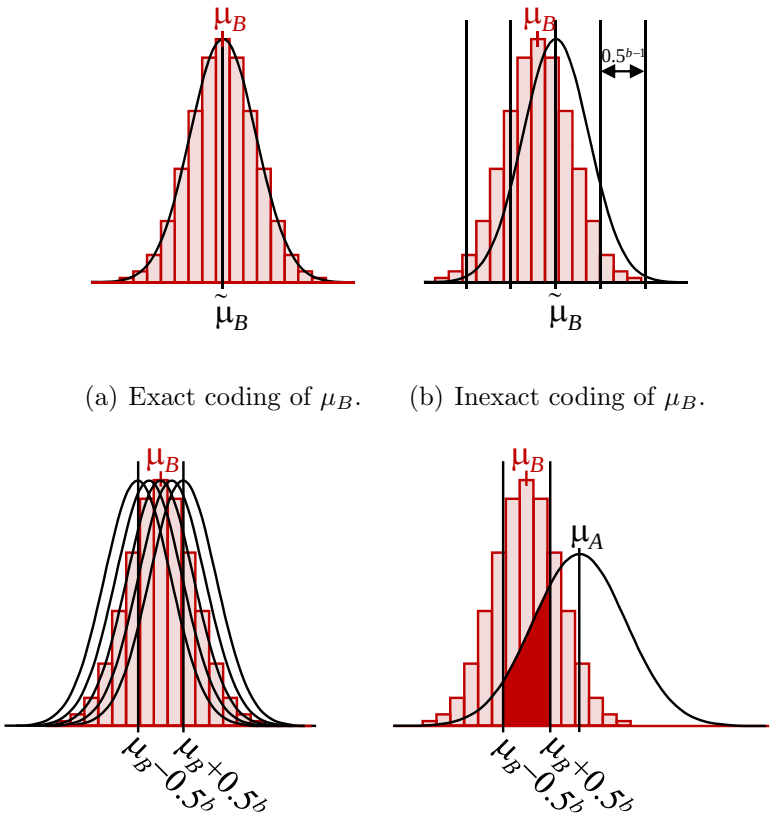
Since every point x is associated to that cluster C in the HCS which has maximum probability density, we can rearrange the terms of the above formula and combine the costs of all points that are in the same cluster:

$$
\begin{aligned}
&= -\sum_{x \in DB} \log_2 \left(W_{Cl(x)} \prod_{1 \leq j \leq d} N(\mu_{Cl(x),j}, \sigma^2_{Cl(x),j}, x_j) \right) \\
&= -\left(\left(\sum_{C \in \mathcal{N}} n W_C \log_2 W_C \right) + \right. \\
&\qquad \left. + \left(\sum_{x \in DB;\, 1 \leq j \leq d} \sum \log_2 N(\mu_{Cl(x),j}, \sigma^2_{Cl(x),j}, x_j) \right) \right) \\
&= -\sum_{C \in \mathcal{N}} \left(n W_C \log_2 W_C + \sum_{x \in C;\, 1 \leq j \leq d} \sum \log_2 N(\mu_{C,j}, \sigma^2_{C,j}, x_j) \right).
\end{aligned}
$$

The ability to model the coding costs of each cluster separately allows us now, to focus on a single cluster, and even on a single dimension of a single cluster.

A common interpretation of the term $-nW_C \log_2 W_C$, which actually comes from the weight a single Gaussian contributes to the GMM, is a Huffman coding of the cluster ID. We assume that every point carries the information which cluster it belongs to, and a cluster with many points gets a shortly coded cluster ID. These costs are referred to the *ID cost* of a cluster C. Lets consider two clusters, A and B, where $B \sqsubseteq A$. We now want to derive the coding scheme for the cluster B and its associated points, given the hierarchical relationship between A and B. Several points are associated with B, where the overall weight of assignment sums up to W_B. When coding the parameters of the associated PDF of B, i.e. μ_B, and σ_B, we have to consider two aspects:

1. The *precision* both parameters should be coded to minimize the overall description length depends on W_B, as well as on σ_B. For instance, if only few points are associated to cluster B and/or the variance σ_B is very large, then it is not necessary to know the position of μ_B very precisely and vice versa.



(a) Exact coding of μ_B.

(b) Inexact coding of μ_B.

(c) Gaussians w.r.t. different grid positions.

(d) All possible values for recovered μ_B.

Figure 3.1: Optimization of the grid resolution for the hMDL criterion.

2. The knowledge of the PDF of cluster A can be exploit for the coding of μ_B, because for likely positions (according to the PDF of A) we can assign fewer bits following the principle of Huffman coding.

Basically, model selection criteria, such as the Bayesian Information Criterion (BIC) or the Aikake Information Criterion (AIC) already address the first aspect, but not the hierarchical aspect (2). In contrast to BIC, which uses the natural logarithm, we use the binary logarithm to represent the code length in *bits*.

For simplicity, we assume that our PDF is univariate and the only parameter is its mean value μ_B. That means, we neglect σ_B by assuming e.g. some fixed value for all clusters. We drop these assumptions at the end of this Section. When the true PDF of cluster B with parameter μ_B is coded inexactly by some parameter $\widetilde{\mu}_B$, the coding costs for each point x (which truly belongs to the distribution $N(\mu_B, \sigma_B^2, x)$) in B is increased compared to the *exact* coding of μ_B, which would result in c_{ex} bits per point:

$$\begin{aligned} c_{\text{ex}} &= \int_{-\infty}^{+\infty} -\log_2(N(\mu_B, \sigma_B^2, x)) \cdot N(\mu_B, \sigma_B^2, x)\ \mathbf{d}x \\ &= \log_2(\sigma_B \sqrt{2\pi \cdot \mathbf{e}}). \end{aligned}$$

If $\widetilde{\mu}_B$ instead of μ_B is applied for compression, we obtain:

$$c(\widetilde{\mu}_B, \mu_B) = \int_{-\infty}^{+\infty} -\log_2(N(\widetilde{\mu}_B, \sigma_B^2, x)) \cdot N(\mu_B, \sigma_B^2, x)\, \mathbf{d}x.$$

The difference is visualized in Figure 3.1(a) and 3.1(b) respectively: In 3.1(a) $\widetilde{\mu}_B$ of the coding PDF, depicted by the Gaussian function, fits exactly to μ_B of the data distribution, represented by the histogram. This causes minimum code lengths for the compressed points but also a considerable effort for the coding of μ_B. In Figure 3.1(b) μ_B is coded by some regular quantization grid. Thereby, the costs for the cluster points slightly increase, but the costs for the location parameter decreases. The difference between $\widetilde{\mu}_B$ and μ_B

depends on the bit resolution and on the position of the quantization grid. One example is depicted in Figure 3.1(b) by five vertical lines, the Gaussian curve is centered by the vertical line closest to μ_B. We derive lower and upper limits of $\widetilde{\mu}_B \in [\mu_B - 1/2^b ... \mu_B + 1/2^b]$ from the number of bits b, spent for coding $\widetilde{\mu}_B$. The real difference between μ_B and $\widetilde{\mu}_B$ depends again on the grid position. Not to prefer clusters that are incidentally aligned with the grid cells, we average over *all* possible positions of the discretization grid. Figure 3.1(c) presents five different examples of the infinitely many Gaussians that could be recovered w.r.t. different grid positions. Note that all positions inside the given interval have equal probability. Hence, the average coding costs for every possible position of $\widetilde{\mu}_B$ can be expressed by the following integral:

$$\begin{aligned} c_{\text{appx}}(b) &= 2^{b-1} \int_{\mu_B - 1/2^b}^{\mu_B + 1/2^b} c(\widetilde{\mu}_B, \mu_B)\ \mathbf{d}\widetilde{\mu}_B \\ &= \frac{1}{2}\log_2(\pi \cdot \mathbf{e} \cdot \sigma_B^2) + \frac{1}{2} + \frac{\log_2 \mathbf{e}}{6\sigma_B^2} \cdot 4^{-b}. \end{aligned}$$

Coding all $n \cdot W_B$ coordinates of the cluster points as well as the parameter μ_B (neglecting the ID cost) requires then the following number of bits:

$$C_{\text{appx}}(B) = c_{\text{appx}}(b) \cdot n \cdot W_B + b.$$

The optimal number b_{opt} of bits is determined by setting the derivation of the above term to zero.

$$\frac{\mathbf{d}}{\mathbf{d}b} C_{\text{appx}}(B) = 0 \implies b_{\text{opt}} = \frac{1}{2}\log_2\left(\frac{n \cdot W_B}{3 \cdot \sigma_B^2}\right).$$

The unique solution to this equation corresponds to a *minimum*, as can easily be seen by the second derivative.

Utilization of the Hierarchical Relationship. We do not want to code

the (inexact) position of μ_B without the prior knowledge of the PDF, associated to cluster A. Without this knowledge, we would have to select a suitable range of values and code μ_B at the determined precision b assuming e.g. a uniform distribution inside this range. In contrast, μ_B is a value taken from the distribution function of cluster A. Hence, the number of bits used for coding of μ_B corresponds to the overall density around the imprecise interval defined by μ_B, i.e.

$$c_{\text{hMDL}}(\mu_B) = -\log_2 \int_{\mu_B - 1/2^b}^{\mu_B + 1/2^b} N(\mu_A, \sigma_A^2, x)\ \mathbf{d}x.$$

Figure 3.1(d) visualizes the complete interval of all possible values for the recovered mean value (marked in red) and illustrates the PDF of the cluster A, which is the predecessor of cluster B. $\widetilde{\mu}_B$ can be coded by determining the whole area under the PDF of A where $\widetilde{\mu}_B$ could be. The area actually corresponds to a probability value. The negative logarithm of this probability represents the required code length for μ_B. The costs for coding all points of cluster B and μ_B then corresponds to

$$c_{\text{appx}}(b) \cdot n \cdot W_B + c_{\text{hMDL}}(\mu_B).$$

Note, that it is also possible to optimize b directly by setting the derivative of this formula to zero. However, this is impossible in an analytic way, and the difference to the optimum which is obtained by minimizing $C_{\text{appx}}(B)$ is negligible. In addition, if the parent A of B is not the root of the HCS, μ_B causes some own ID cost. In this case, μ_B is a sample from the complex distribution function of the hierarchical cut (cf. Definition 2), which prunes the complete level of B and all levels below. Hence, the weight of these levels is added to the new leaf nodes (after cutting), and the ID costs of μ_B correspond to:

$$-\log_2 \left(\sum_{X \sqsubseteq A} W_X\right).$$

A similar analysis can be done for the second parameter of the distribution function, σ_B. Since it is not straightforward to select a suitable distribution function for the Huffman coding of variances, one can apply a simple trick: Instead of coding σ_B, we code $y_B = \mu_B \pm v \cdot \sigma_B$, where v is a constant close to zero. Then, y_B is also a sample from the distribution function $N(\mu_A, \sigma_A^2, x)$ and can be coded similar to μ_B. Therefore, $c_{\text{hMDL}}(\sigma_B) = c_{\text{hMDL}}(\mu_B)$, and we write $c_{\text{hMDL}}(\text{param})$ for the coding costs per parameter instead. In general, if the PDF, which is associated with a cluster has r parameters, then the optimal number of bits can be obtained by the formula:

$$b_{\text{opt}} = \frac{1}{2} \log_2(\frac{n \cdot W_B}{3 \cdot r \cdot \sigma_B^2}).$$

And the overall coding costs are:

$$C_{\text{hMDL}}(B) = c_{\text{appx}}(b) \cdot n \cdot W_B + r \cdot c_{\text{hMDL}}(\text{param})$$

Until now, only the trade-off between coding costs of points and the parameters of the assigned cluster are taken into account. If we go above the lowest level of the HCS, we have to trade between coding costs of parameters at a lower level and coding costs of the parameters at the next higher level. This can be done in a similar way as before: Let b_B be the precision, which has already been determined for the representation of μ_B and σ_B, the parameters for cluster B, which is a subcluster of A. However, this is the minimum coding costs assuming that μ_A and σ_A have been stored at maximum precision, and that μ_B and σ_B are also given. Now, we assume that μ_B is an arbitrary point selected from the distribution function $N(\mu_A, \sigma_A^2, x)$ and determine an expectation for the cost:

$$\int_{-\infty}^{+\infty} -\log_2 \int_{\mu_B - 1/2^{b_B}}^{\mu_B + 1/2^{b_B}} N(\mu_A, \sigma_A^2, x) \mathbf{d}x \; N(\mu_A, \sigma_A^2, \mu_B) \mathbf{d}\mu_B.$$

Finally, we assume that μ_A is also coded inexactly by its own grid with resolution b_A. Then the expected costs are:

$$2^{b_A-1}\int_{\mu_A-1/2^{b_A}}^{\mu_A+1/2^{b_A}}\int_{-\infty}^{+\infty}\left(-\log_2\int_{\mu_B-1/2^{b_B}}^{\mu_B+1/2^{b_B}} N(y,\sigma_A^2,x)\,\mathbf{d}x\right)\cdot$$

$$\cdot\, N(\mu_A,\sigma_A^2,\mu_B)\,\mathbf{d}\mu_B\,\mathbf{d}y.$$

Since it is analytically impossible to determine the optimal value of b_A, we can easily get an approximation of the optimum by simply treating μ_B and σ_B like the points which are directly associated to the cluster A. The only difference is the following. While the above integral considers that the PDF varies inside the interval $[\mu_B - 1/2^{b_B}, \mu_B + 1/2^{b_B}]$ and determines the average costs in this interval, treating the parameters as points only considers the PDF value at one fixed position. This difference is negligible provided that $\sigma_B < \sigma_A$, which makes sense as child clusters should usually be much smaller (in terms of σ) than their parent cluster.

Coding Costs for a Cluster. Summarizing, the coding costs for a cluster can be obtained as follows:
(1) Determine the optimal resolution parameter for each dimension according to the formula:

$$b_{\text{opt}} = \frac{1}{2}\log_2\left(\frac{n\cdot W_B + r\cdot \#\text{ChildNodes}(B)}{3\cdot r\cdot \sigma_B^2}\right).$$

(2) Determine the coding costs for the data points and the parameters according to:

$$C_{\text{hMDL}}(B) = c_{\text{appx}}(b)\cdot n\cdot W_B + r\cdot c_{\text{hMDL}}(\text{param})$$

(3) Add the costs obtained in step (2) to the ID costs of the points ($-nW_B\log_2(W_B)$) and of the parameters ($-\log_2(\sum_{X\sqsubseteq A} W_X)$). Whereas the costs determined

in (2) are individual in each dimension the costs in (3) occur only once per stored point or parameter set of a cluster.

Coding Costs for the HCS. The coding costs for all clusters sum up to the overall coding costs of the hierarchy where we define constant ID costs for the parameters of the root:

$$hMDL = \sum_{C \in \mathcal{N}} \left(C_{\text{hMDL}}(C) - nW_C \log_2(W_C) - \log_2(\sum_{X \sqsubseteq \text{parent of } C} W_X) \right).$$

3.2.4 Algorithm ITCH

This Section is devoted to efficient heuristics for obtaining and optimizing the HCS according to hMDL. Basically, we optimize our objective function in an EM-like clustering algorithm ITCH where all clusters compete for data points. Reassignment of objects and re-estimation of the parameters of the HCS are done interchangeably until convergence. Additionally, starting from a suitable initialization, ITCH periodically modifies the HCS by two operations, *delete* and *collapse*.

Initialization of the HCS. Clustering algorithms that follow the EM-scheme have to be suitable initialized before starting with the actual iterations of E-step and M-step. An established method for the flat EM algorithm is to initialize with the result of a k-means clustering. This is typically repeated several times with different seeds and the result with best mean squared overall deviation from the cluster centers is taken. Following this idea, ITCH uses a initialization hierarchy determined by a bisecting k-means algorithm taking the hMDL value of the HCS as a stopping criterion for partitioning. First, a root node that contains all points is created. Then this root node is partitioned into two subclusters by applying k-means with $k = 2$. This is done recursively until the hMDL of the binary HCS does

not improve anymore within three steps. This ensures not to get stuck in a local minimum. Finally, after the best hierarchy is selected, μ_C and Σ_C are determined for each node C according to Section 3.2.3, and equal weights are assigned to the nodes.

E-step and M-step. As indicated before, the points can be assigned directly and indirectly to the clusters of the HCS. Whenever a point is associated directly to a cluster C then it is also indirectly associated to every ancestor of C in the HCS. Nevertheless, points can also be directly associated not only to leaf nodes but also to inner nodes of the HCS. For instance, if a point p_i is an outlier that does not fit to any of the most specific clusters at the bottom level of the hierarchy, then p_i has to be associated to an inner node or even the root of the HCS.

As established in Section 3.2.2, the clusters at all levels of the HCS compete for the data points. A point x is directly associated to that Cluster $C \in \mathcal{N}$ the probability density function of which is maximal:

$$Cl(x) = \arg\max_{C \in \mathcal{N}} \{W_C \cdot N(\mu_C, \Sigma_C, x)\}.$$

In the E-step of our hierarchical clustering algorithm, the direct association $Cl(x)$ for every object x is updated. Whereas, in the E-step only the direct association is used (or updated) in the M-step which updates the location and scale parameters of all clusters we use both the direct and indirect association. The motivation is the following: The distribution function of every node in the HCS should always represent the whole data set in this branch of the tree, and the root node should even represent the complete data set. Therefore, for the location and scale parameters, all directly and indirectly associated objects are considered, as in the following formulas:

$$\begin{aligned}\mu_C &= \frac{\sum_{B\in\mathcal{N},B\sqsubseteq C}\left(\sum_{x\in B} x\right)}{\sum_{B\in\mathcal{N},B\sqsubseteq C}|B|}\\ \sigma^2_{C,j} &= \frac{\sum_{B\in\mathcal{N},B\sqsubseteq C}\left(\sum_{x\in B}(x_j-\mu_{C,j})^2\right)}{\sum_{B\in\mathcal{N},B\sqsubseteq C}|B|}\\ \Sigma_C &= \mathrm{diag}(\sigma^2_{C,1},...,\sigma^2_{C,d}).\end{aligned}$$

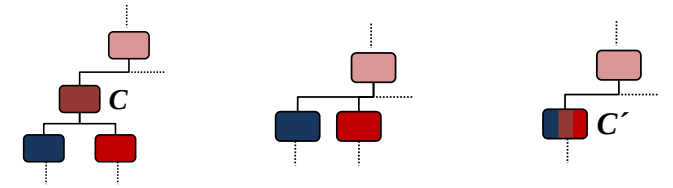
In contrast, the weights W_C of each cluster in the HCS should reflect how strong the individual Gaussian in the overall mixture of the HCS is and sum up to 1 in order to define a valid PDF with an integral over the complete data space of 1. Therefore, we only apply the direct association to the update rule of the cluster weight, and we obtain:

$$W_C = |C|.$$

Rearrangement of the HCS. The binary HCS that results from the initialization step does not limit our generality. ITCH is flexible enough to reconstruct the HCS and thereby to change it into a general one. Given a binary hierarchy which is deeper than any n-ary hierarchy with $n > 2$ ITCH aims in flattening the HCS as far as the rearrangement improves our hMDL criterion.
Therefore we trade off the following operations to eliminate clusters that do not pay off any more:

- *delete* one node or
- *collapse* a node with all child nodes.

Figure 3.2 visualizes the operations for an extract of a given HCS. By deleting a cluster C from the HCS the child nodes of C become child nodes of the parent of C (Figure 3.2(b)). By collapsing C, all of its child nodes are merged into a new cluster C' (including C), and therefore all of their child nodes



(a) Initial state. (b) Deletion. (c) Collapse.

Figure 3.2: Restructuring operations of ITCH.

become child nodes of C' (Figure 3.2(c)). Afterwards, all points are redistributed and E-step and M-step are performed alternately until convergence. Thus, the point to cluster assignment expressed by the weights may change. ITCH processes the rearrangement in an iterative way. In each iteration it tentatively deletes and collapses each node in the HCS and performs E- and M-steps. The node and the operation that improves the hMDL criterion best, lets call it *winner cluster*, is selected and then the local neighborhood (parent, child and sibling nodes) is processed. These steps, detecting the *winner cluster* and processing the local neighborhood, are performed until convergence.

3.2.5 Experiments

This Section provides an extensive experimental evaluation on synthetic and real-world data sets. In all experiments, the input parameters of all methods have been optimized in terms of quality and the best results have been reported in order to achieve a fair comparison. Since ITCH is a hybrid approach combining the benefits of hierarchical, model-based and parameter-free clustering, we compare to algorithms of these classes to demonstrate the effectiveness of ITCH. More precisely, we selected the hierarchical clustering method Single Link [153], OPTICS [4], a more outlier-robust hierarchical clustering algorithm, and RIC [18] an outlier-robust and parameter-free state-of-the-art

algorithm to model-based clustering. Class labels are assigned to the data which was used only for evaluation but not for clustering. To relieve evaluation w.r.t. outliers, we added a color bar below the dendrograms of Single Link and the reachability plots of OPTICS, where colors refer to the class labels in the original data. A quantitative comparison is performed using the quality measures described in Section 2.2.

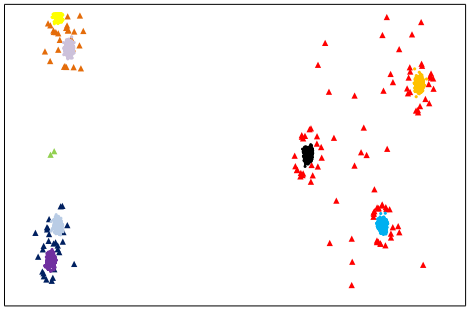
Synthetic Data

First, ITCH has been evaluated on several synthetic data sets. Exemplary, the results on two data sets named DS_1 and DS_2 are shown. Data set DS_1 comprises 3,662 2-dimensional points that form a hierarchy of 12 clusters. At the bottom level, seven Gaussian clusters are located. The data set contains several outliers at different levels of the hierarchy, cf. Figure 3.3(a).

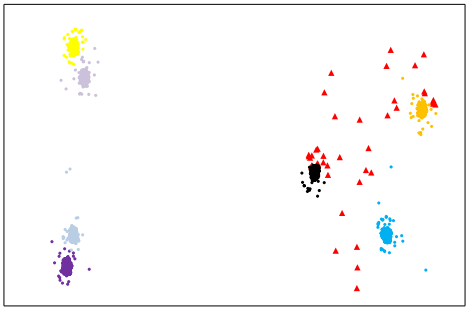
DS_2 is a non-hierarchical data set that is composed of two Gaussian clusters with 1,650 points each, overlapping in the marginal area without any global outliers (cf. Figure 3.4(a)).

Table 3.1: Quantitative evaluation on DS_1.

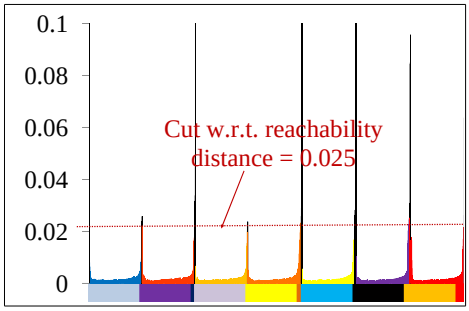
	ITCH	RIC	OPTICS	Single Link
NMI	0.9895	0.8665	0.9585	0.9555
AMI	0.9894	0.8453	0.9461	0.9632
Prec	0.9958	0.8589	0.9654	0.9903
Rec	0.9956	0.8883	0.9719	0.9601
DOM	0.1334	0.4279	0.2167	0.2142



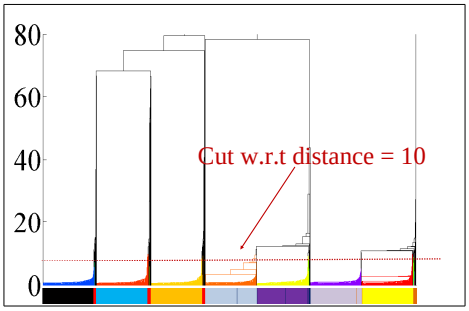
(a) Synthetic data set DS_1.



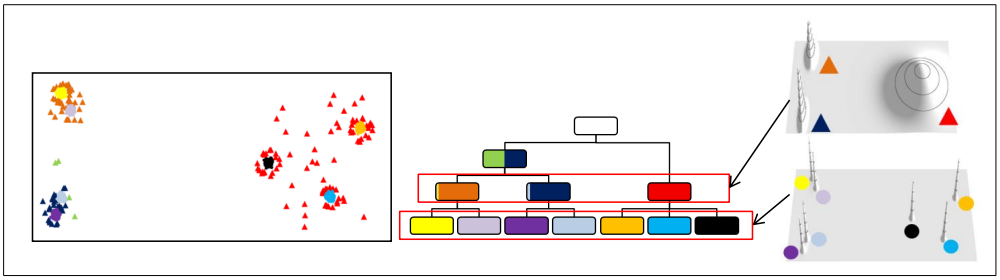
(b) Result of RIC.



(c) Result of OPTICS.



(d) Result of Single Link.



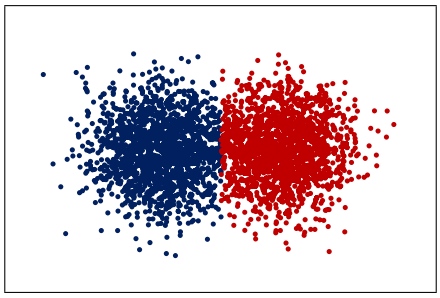
(e) Result of ITCH including the PDFs for each cluster at level 0 and 1.

Figure 3.3: Experimental evaluation on synthetic data set DS_1.

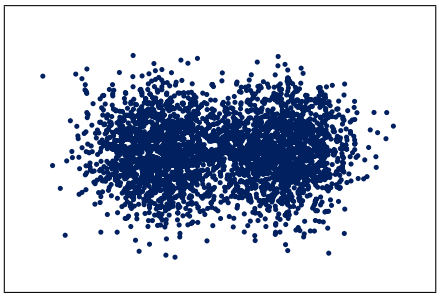
Experimental Evaluation on DS_1. In order to apply RIC to the hierarchical data set, we preprocessed DS_1 with Single Link and applied RIC as postprocessing step. Figure 3.3(b) demonstrates the result of RIC (Precision: 85.89% Recall: 88.88%). It is obvious that RIC fails to successfully filter out all outliers. More precisely, RIC assigns points (marked by a dark blue and orange triangle in the original data) that obviously are outliers w.r.t. two clusters on the left upper and lower side misleadingly to clusters. Also a majority of the red outliers are incorrectly identified as cluster points. Figure 3.3(c) shows the result of OPTICS on DS_1 with a cut at reachability distance = 0.025. As Clusters can be recognized as valleys in the reachability plot, OPTICS yields a satisfactory result (Precision: 96.54% Recall: 97.19%). Also, the hierarchy produced by Single Link (Figure 3.3(d)) reflects the true hierarchy quite well. A cut through the hierarchy at distance = 10 produces seven disjoint clusters. However, in terms of accuracy Single Link and OPTICS show worse results than ITCH (cf. Table 3.1). ITCH is the best method to detect the true cluster hierarchy including outliers fully automatically (Precision: 99.58% Recall: 99.56%), and ITCH provides meaningful models on the data for each level of the hierarchy (Figure 3.3(e)). Futhermore, considering all quality measures, the result of ITCH is best in terms of effectiveness.

Table 3.2: Quantitative evaluation on DS_2.

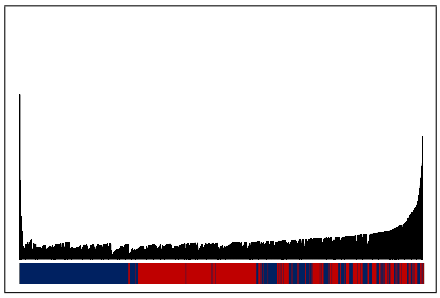
	ITCH	RIC	OPTICS	Single Link
NMI	0.9586	0.0000	0.0000	0.0000
AMI	0.9586	0.0000	0.0000	0.0000
Prec	0.9949	0.2521	0.2521	0.5000
Rec	0.9948	0.5021	0.5021	1.0000
DOM	0.0332	0.6956	0.6956	0.6956



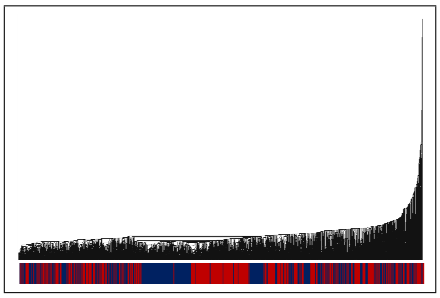
(a) Synthetic data set DS_2.



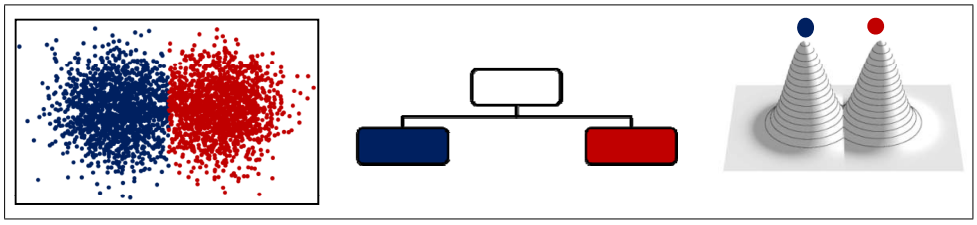
(b) Result of RIC.



(c) Result of OPTICS.



(d) Result of Single Link.



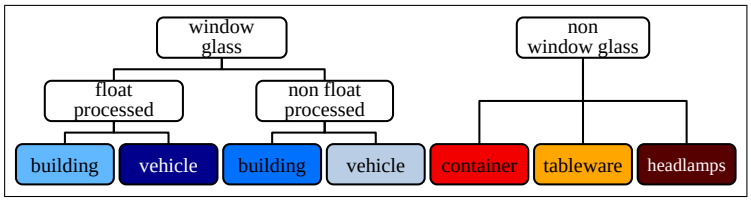
(e) Result of ITCH including the PDFs for both clusters at level 0.

Figure 3.4: Experimental evaluation on synthetic data set DS_2.

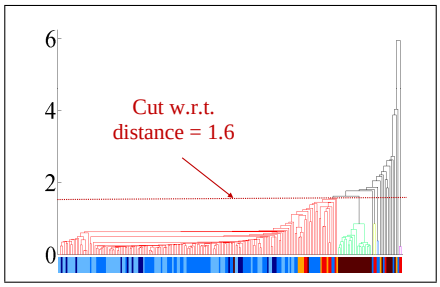
Experimental Evaluation on DS_2. Figure 3.4(b) demonstrates that RIC merges the two Gaussian clusters into only one cluster. Also with OPTICS, it is impossible to detect the true structure of DS_2. The color bar in Figure 3.4(c) indicates that OPTICS assigns the points in an almost arbitrary order. Even when increasing the parameter for the minimum object density per cluster to a large value, OPTICS fails in detecting two clusters. Single Link miscarries due to the massive Single Link effect (Figure 3.4(d)). Here, OPTICS is not suitable to cure that problem. Moreover, the hierarchies generated by OPTICS and Single Link are overly complex but do not capture any cluster structure. Hence, for quantitative evaluation we assigned all points to one cluster. Only ITCH discovers a meaningful result without requiring any input parameters (Precision: 99.49% Recall: 99.48%). All clusters that do not pay off w.r.t. our hMDL are pruned and hence, only two Gaussian clusters remain which are described by an intuitive description in form of a PDF (Figure 3.4(e)). The quantitative comparison of all methods is given in Table 3.2.

Table 3.3: Quantitative evaluation on glass data.

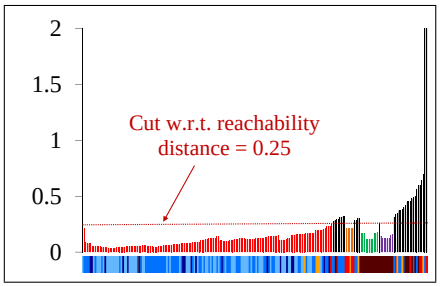
	ITCH	RIC	OPTICS	Single Link
NMI	0.3390	0.1221	0.4029	0.4110
AMI	0.3222	0.0804	0.3092	0.3085
Prec	0.5350	0.1710	0.4739	0.3812
Rec	0.3551	0.1822	0.4626	0.4393
DOM	1.4805	1.5872	1.2980	1.5010



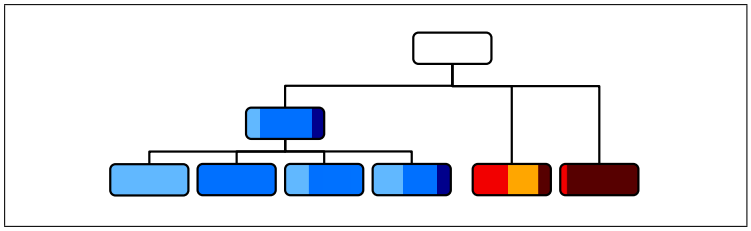
(a) Hierarchy in the original data set.



(b) Result of Single Link.



(c) Result of OPTICS.



(d) Result of ITCH.

Figure 3.5: Hierarchical clustering of 9-dimensional glass data (214 instances). Figure 3.5(a) provides information about the number of instances, which are associated with the particular classes in the original data set.

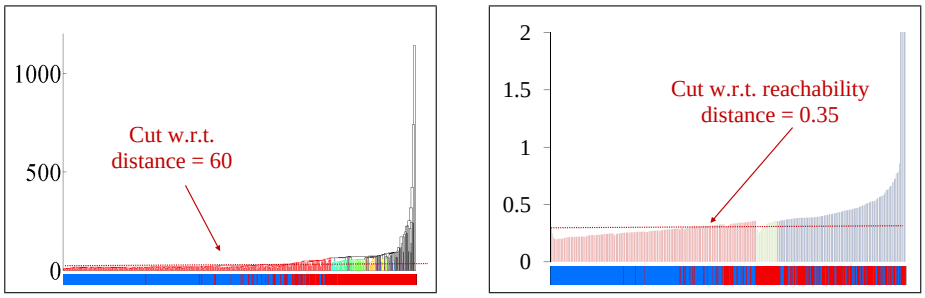
Real World Data

In addition to the synthetic data sets, the effectiveness of ITCH has been evaluated by using several real-world data sets available at UCI [1].

Glass Data. The *Glass Identification* data set comprises nine numerical attributes representing different glass properties. 214 instances are labeled according to seven different types of glass that form a hierarchy as presented in Figure 3.5(a). ITCH perfectly separates *window glass* from *non window glass*. Additionally, *tableware* and *containers* are almost perfectly separated from *headlamps*. ITCH arranges the subclusters of *window glass* in the same subtree. Some outliers are directly assigned to the cluster *window glass*. In contrast to ITCH, neither Single Link nor OPTICS separates *window glass* from *non window glass* perfectly as can be seen from the color bars below the plots (cf. Figures 3.5(b) and 3.5(c)). *Containers* and *tableware* do not form discrete clusters but are constituted as outliers instead. In the dendrogram only the *headlamps* can be identified clearly, whereas in the reachability plot this cluster is split into two clusters. Nevertheless, both approaches do not reflect the original hierarchy successfully. As it is not clear where to define an adequate cut through the dendrogram we applied RIC at the bottom level. This results in only two clusters without any separation between *window* or *non window glass*. Table 3.3 summarizes the quantitative evaluation on this data set. For OPTICS, the results refer to the partitioning clustering with a maximum reachability distance of 0.25. For Single Link, a cut through the dendrogram at distance = 1.6 was used for the computation. In terms of precision, ITCH outperforms all other methods.

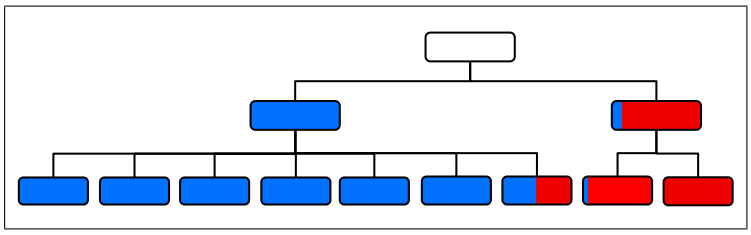
Cancer Data. The *Breast Cancer Wisconsin* data set contains 569 instances each describing 30 different characteristics of the cell nuclei present in an digitized image of a breast cancer mass. There are two classes *benign*

[1] `http://archive.ics.uci.edu/ml/`



(a) Result of Single Link.

(b) Result of OPTICS.



(c) Result of ITCH.

Figure 3.6: Hierarchical clustering of 30-dimensional breast cancer data (569 instances).

Table 3.4: Quantitative evaluation on cancer data.

	ITCH	RIC	OPTICS	Single Link
NMI	0.3832	0.4595	0.2835	0.3097
AMI	0.2044	0.3612	0.2518	0.2127
PREC	0.9574	0.9846	0.8033	0.8884
REC	0.2830	0.6608	0.7575	0.6626
DOM	0.2635	0.3022	0.4752	0.4771

and *malignant*, represented by the colors blue and red, respectively. Both, the dendrogram of Single Link, and the reachability plot of OPTICS show a massive chaining effect (see Figures 3.6(a) and 3.6(b)). Even though, a cut at reachability distance = 0.35 in the OPTICS plot results in three clusters. However, no real cluster structure is visible within the clusters. The same holds for the Single Link dendrogram. The color bars below the dendrogram and the OPTICS plot indicate that both methods fail to separate *benign* from *malignant* objects. For separating the two classes we applied RIC on top of a k-Means clustering with k=15. However, RIC also fails to separate the two classes and results in three mixed clusters with objects of class *benign* and *malignant*. In contrast, despite the high dimensionality of the data, ITCH almost perfectly separates the *benign* from the *malignant* cells which are then split into different subclusters (cf. Figure 3.6(c)). Table 3.4 gives the quantitative evaluation of the results with a cutoff value = 60 for Single Link and a cut w.r.t. reachability distance = 0.35 for OPTICS.

Stability of ITCH

Since we do not want to rely on single results we additionally tested the stability of ITCH over 20 runs for each data set. Figure 3.7 shows the variance

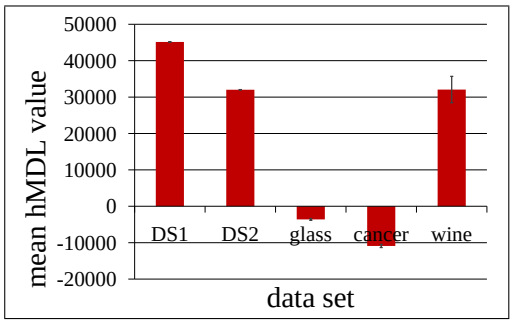


Figure 3.7: Stability of the ITCH result over 20 runs.

of the hMDL value in percent depending on the mean value. The result of ITCH is highly stable within DS_1 and DS_2 having only a variance of 0.03% and 0.12%, respectively. Also in the real world data sets the result of ITCH shows only little variance.

3.3 Genetic Algorithm for Finding Cluster Hierarchies

As mentioned before the result of many existing algorithms for hierarchical clustering heavily depend on their initialization or vary with different parameter settings. Moreover, the identification of outliers and assignment to adequate levels of the hierarchy is not considered properly by all methods. ITCH is the only method that can cope with all these problems so far (cf. Chapter 3.2). However, ITCH suffers from the following drawback. It uses an optimization heuristic that inhibits the algorithm to exhaustively search the space to find the correct cluster structure.

Many methods have been proposed that are based on a genetic algorithm to solve the NP-hard clustering problem [99, 46, 112, 152, 109, 136]. A genetic algorithm (GA) is a stochastic optimization technique based on the mechanism of natural selection and genetics, originally proposed by [81]. The general idea behind a GA is that the candidate solutions to an optimization problem (called *individuals*) are often encoded as binary strings (called *chromosomes*). A collection of these chromosomes forms a *population*. The evolution initially starts from a random population that represents different individuals in the search space. In each generation, the fitness of every individual is evaluated, and multiple individuals are then selected from the current population based on Darwin's principle "Surviving of the fittest". These individuals build the mating pool for the next generation. The new population is then formed by the application of recombination operations like *crossover* and *mutation*. A GA commonly terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

However, all these methods mentioned before are non-hierarchical clustering methods. Therefore, the hierarchical clustering algorithm GACH (Ge-

netic Algorithm for Finding Cluster Hierarchies) is proposed in this Chapter that is based on a stochastic optimization technique. GACH overcomes the limitations of existing hierarchical clustering approaches mentioned before and claims the following contributions:

- *Fitness*: The fitness of different chromosomes is optimized using a MDL-based optimization technique.

- *No difficult parameter-setting*: Besides the parameters that are specific for a genetic algorithm, GACH requires no expertise about the data (e.g. the number of clusters).

- *Flexibility*: By the use of a GA-based stochastic search GACH thoroughly explores the search space and is therefore flexible enough to find the correct hierarchical cluster structure and is not sensitive to the initialization.

- *Outlier-robust*: Outliers are assigned to the root of the cluster hierarchy or to an appropriate inner node, depending on the degree of outlierness.

- *Model description*: The content of each cluster is described by a PDF.

Section 3.3.1 introduces the basic definitions of a genetic algorithm and explains the necessary modifications to use a GA for cluster hierarchies. Section 3.3.2 introduces the main concepts of the new GACH algorithm. An extensive evaluation of GACH is provided in Section 3.3.3. First, the choice and impact of the genetic parameters are discussed and then, an extensive experimental evaluation shows that GACH outperforms state-of-the art hierarchical clustering methods using synthetic and real data sets.

3.3.1 Using Genetic Algorithm for Finding Cluster Hierarchies

Each chromosome specifies one solution to a defined problem. For GACH, a chromosome is the encoding of a hierarchical cluster structure (HCS) that was previously defined in ITCH (Section 3.2.2) and has to address the three following features:

- Storage of a set of clusters $C_1, \cdots, C_n$.
- Representation of the hierarchical relationship between clusters forming a tree $\mathcal{T}$ of clusters.
- Encoding of the cluster representatives, i.e. the parameters of the underlying PDF. For GACH we represent each cluster by a Gaussian PDF. Note that our model can be extended to a variety of other PDFs, e.g. uniform or Laplacian.

With this requirements a chromosomal representation of a HCS is defined as follows:

Definition 3 (Chromosomal HCS)
(1) A chromosomal HCS (HCS_{Chrom}) is a dynamic list storing a set of cluster objects.
(2) Each cluster C holds references to its parent cluster and to its child nodes. Besides that, the level l_C of each cluster defines the height of the descendant subtree in the HCS
(3) The parameters of the underlying Gaussian PDF of cluster C, the mean value μ_C and σ_C, are modeled as additional parameters of the cluster C.
(4) Each cluster C is associated with a weight W_C, where $\sum_{i=0}^{k-1} W_{C_i} = 1$.

The underlying PDF of a cluster C is a multivariate Gaussian in a d-dimensional data space which is defined by the parameters μ_C and σ_C (where μ_C and σ_C

are vectors from a d-dimensional space) by the following formula:

$$N(\mu_C, \sigma_C, x) = \prod_{1 \leq i \leq d} \frac{1}{\sqrt{2\pi\sigma_{C,i}^2}} \cdot \mathbf{e}^{-\frac{(x_i - \mu_{C,i})^2}{2\sigma_{C,i}^2}}$$

GACH assigns each point x *directly* to that cluster $C \in HCS_{Chrom}$ the probability density of which is maximal at the position of x:

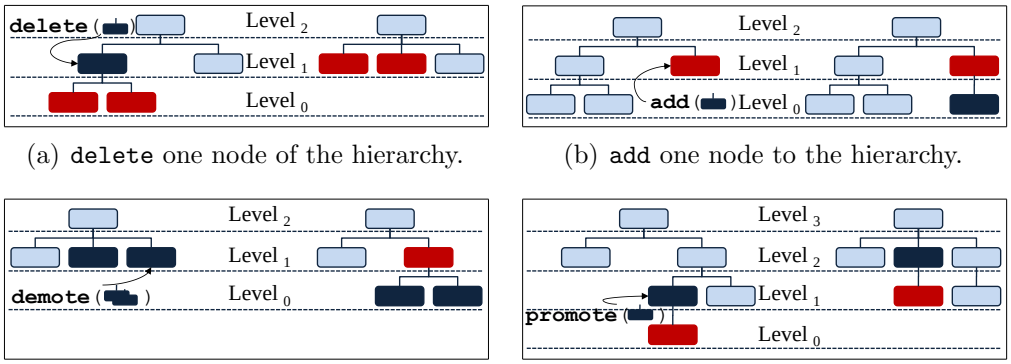
$$C(x) = \arg \max_{C_i \in HCS_{Chrom}} \{W_{C_i} \cdot N(\mu_{C_i}, \sigma_{C_i}, x)\}.$$

The parameters μ_C and σ_C of each cluster C are determined based on the hierarchical relationship to all subclusters and calculated analogously to ITCH as described in Section 3.2.4.

Initialization of GACH. Basically the initial set of a population consists of a randomly generated set of individuals. This strategy is also processed by GACH, where in a first step a random number of clusters $\tilde{k}$ is selected for each structure HCS_{Chrom}. Then a simple k-means algorithm divides the data set into $\tilde{k}$ clusters that act as the leafs of the initial hierarchy. Finally, these clusters are combined by one additional root cluster. Hence, the initialization process results in a 2-level hierarchy that consists of $\tilde{k} + 1$ nodes. Each cluster C_i is described by random parameters and is associated a weight $W_{C_i} = \frac{1}{k}$.

Reproduction. In order to generate the next population of cluster hierarchies GACH uses several genetic operators that are particularly defined for the hierarchical clustering problem: mutations (`delete`, `add`, `demote` and `promote`) and `crossover`.

The `delete` operator deletes a specific cluster C (except the root) with a deletion rate p_{del} from the HCS. This results in structure HCS$'$ that does not contain the cluster C any more. The proceeding of `delete` is illustrated in Figure 3.8(a). Here, the cluster C is marked in dark blue color. By deleting



(a) `delete` one node of the hierarchy.

(b) `add` one node to the hierarchy.

(c) `demote` two nodes to a lower level.

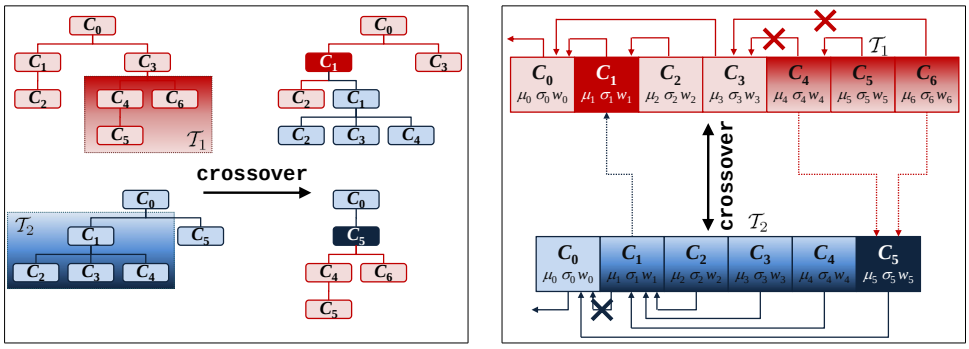
(d) `promote` one node to a higher level.

Figure 3.8: Summarization of the mutation operators used for GACH.

C the level of each direct and indirect subcluster of C (marked in red) is decreased by 1. The former parent node of C, the root node in our example, becomes the parent node of all direct subclusters of C.

The operator `add` adds direct subclusters to an arbitrary cluster C of the hierarchy with an add rate p_{add} (normally $p_{del} = p_{add}$). The number of additional subclusters is bounded by an upper limit value max_{new}. Figure 3.8(b) illustrates an example for the application of the `add` operator to a HCS where the cluster C_{add} marked in dark blue color is added as a subcluster of the red cluster C. Since the PDFs of subcluster C_{add} should fit into the PDF of C and therefor to get an valid hierarchical relationship between C and C_{add}, we calculate random parameters for C_{add} based on μ_C and σ_C. In particular, we add a random value r to both parameters, where r is a vector from a d-dimensional space: $\mu_{C_{add}} = \mu_C + r \qquad \sigma_{C_{add}} = \sigma_C + r$.

The motivation behind the `demote` operator is the following. Assume a data set consisting of three clusters C_1, C_2 and C_3, where C_1 holds a large number of objects, clusters C_2 and C_3 are smaller ones but they are locally close to



(a) Hierarchical representation. (b) Chromosomal representation.

Figure 3.9: The `crossover` operator for two selected hierarchies. The subtree $\mathcal{T}_1$ of the red hierarchy is exchanged with the subtree $\mathcal{T}_2$ of the blue hierarchy, visualized by a hierarchical (3.9(a)) and a chromosomal representation (3.9(b)).

each other. An intuitive hierarchical representation would be a HCS with one root node and C_1, C_2 and C_3 as direct subclusters (cf. Figure 3.8(c)) which provides only a very coarse view of the data set. But, if we combine the two smaller clusters (marked in dark blue) and demote them with a demote rate p_{dem} to a lower level with a common parent cluster (marked in dark red), we are able to get a more detailed look on our data. The parameters of the inserted cluster C_{in} are obtained by the average of the parameters of the demoted clusters. Note that demoting only one cluster corresponds to the `add` operator. Hence, we apply `demote` on at least two clusters.

The `promote` operator lifts a cluster C from a lower level to the level right above with a promotion rate p_{pro}, if and only if C is at least two levels underneath the root cluster. Consequently all subclusters of C are lifted accordingly. In Figure 3.8(d) the dark blue cluster is promoted from level 3 to level 2. Hence, also the red subcluster is lifted to the next higher level. The parent of the parent node of the dark blue cluster (here the root node)

becomes the parent node of C in the resulting hierarchy HCS′, together with the correct rearrangement of all subclusters.

The operator `crossover` exchanges information among two different structures. In general the information of two different chromosomes is combined in order to obtain a new individual with superior quality. GACH performs a crossover between two selected hierarchies HCS_1 and HCS_2 with a crossover rate p_{co} as follows:

1. Remove a selected subtree $\mathcal{T}_1$ entirely from HCS_1.
2. Remove a selected subtree $\mathcal{T}_2$ entirely from HCS_2.
3. Select a random node in HCS_1 and insert $\mathcal{T}_2$.
4. Select a random node in HCS_2 and insert $\mathcal{T}_1$.

Figure 3.9(a) illustrates this procedure exemplarily for two selected hierarchies. The subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ are removed from the red and the blue HCS respectively. $\mathcal{T}_1$ is then inserted into the blue HCS as subtree of the dark blue node. Analogously $\mathcal{T}_2$ is inserted as subtree of the dark red cluster in the red HCS. Figure 3.9(b) describes the same procedure w.r.t. a chromosomal representation of both hierarchies. For simplicity, only the pointers to the parent cluster are displayed.

Fitness Function. Following the Darwin‘s principle “Survival of the fittest” naturally only individuals with highest fitness can survive and those that are weaker become extinct. A GA adopts this aspect of evolution by the use of a fitness function. GACH uses the $hMDL$ criterion formalized in Section 3.2.1 which evaluates the fitness of a chromosomal HCS by relating the clustering problem to that of data compression by Huffman Coding:

$$hMDL_{HCS} = \sum_{C \in HCS} \left(cost(C) - nW_C \log_2(W_C) - \log_2(\sum_{x \sqsubseteq \text{parent of } C} W_x) \right)$$

The coding cost for each cluster $C \in HCS$ is determined separately and summed up to the overall coding cost of the complete HCS. Points that are directly assigned to the cluster C together with the parameters μ_C and σ_C of the underlying Gausssian PDF are coded by $cost(C)$. The point to cluster assignment is coded by the so-called ID cost of each data point $x \in C$ and is given by $-nW_C \log_2(W_C)$ where W_C is the weight of cluster C and n the number of points. The binary logarithm is used to represent the code length in bits. Clusters with higher weight are coded by a short code pattern whereas longer code patterns are assigned for smaller clusters with lower weight. The ID costs for the parameters are formalized by $-\log_2(\sum_{x \sqsubseteq \text{parent of } C} W_x)$ whereas constant ID costs are defined for the parameters of the root node.

The better the statistical model (the HCS) fits to the data the higher the compression rate thus the lower the coding costs are. Using this coding scheme as fitness function ensures the selection of that chromosome HCS_{Chrom} that fits best to the data.

Selection. The selection function chooses the best individuals out of a set of given individuals to form the offspring population according to their fitness. For GACH we use the well-known weighted roulette wheel strategy [117]. Imagine that each HCS_{Chrom} represents a number on a roulette wheel, where the amount of numbers refers to the size of the population. In addition we assign a weight to each number on the roulette wheel, depending on the fitness of the underlying chromosome. That means the better the fitness of a chromosome the higher its weight on the roulette wheel will be, i.e. the higher the chance to get selected for the offspring population. Note that there is the chance that one chromosome is selected multiple times. GACH forms

a new population that has as much individuals as the former population.

3.3.2 Algorithm GACH

The algorithm GACH is based on the combination of a genetic algorithm, information theory and model-based clustering as described in the previous Section. An initial population is built as described in Section 3.3.1. This population is evaluated according to the fitness function $hMDL_{HCS}$ which means that GACH determines the coding cost for each cluster hierarchy of the population. The lower the coding costs the better the HCS fits to the data. In order to optimize the point to cluster assignment of each HCS and to provide an additional model of the data, we apply the same hierarchical E- and M-steps formalized in ITCH (cf. Section 3.2.4) on each cluster structure.

The formalization of GACH is presented in Algorithm 1. The population resulting from the initialization undergoes several mutation and crossover operations within pop_{max} number of generations in an iterative way. In each iteration the next population is selected according to the weighted roulette wheel strategy and undergoes several reproduction procedures as described in the previous Section. Each operation (mutation or crossover) is processed with a certain probability which is extensively evaluated in Section 3.3.3. After optimizing the point to cluster assignment using E- and M-step as described in ITCH (cf. Section 3.2.4), GACH determines the fitness of each HCS_{Chrom} in the population by calculating the $hMDL_{HCS}$ value. The algorithm terminates if a specified maximum number of new populations pop_{max} is reached. The experiments show that the HCS can be optimized even with small generation sizes.

3.3.3 Experiments

Now we demonstrate that the genetic parameters (mutation rate, crossover rate and population size) do not affect the effectiveness of GACH in a major

Algorithm 1 GACH

1: $count_{pop} \leftarrow 0$
2: initialize $population(count_{pop})$
3: evaluate $population(count_{pop})$
4: **while** $(count_{pop} \leq pop_{max})$ **do**
5: $count_{pop} \leftarrow count_{pop} + 1$
6: select $population(count_{pop})$ from $population(count_{pop} - 1)$
7: reproduce $population(count_{pop})$
8: evaluate $population(count_{pop})$
9: **end while**

way. Nevertheless, we provide a suitable parametrization that enables the user to receive good results independent of the used data set. Based on this, we compare the performance of GACH to several representatives of various clustering methods on synthetic and real world data. We selected the hierarchical clustering method Single Link [153], the more outlier-robust hierarchical clustering algorithm OPTICS [4], with optimal parameters w.r.t. accuracy. Furthermore, we chose RIC [18], an outlier-robust and information-theoretic clusterer, and finally ITCH (cf. Section 3.2). As ITCH strongly depends on its initialization, we used the best out of 10 runs in this case. In order to facilitate interpretation of the clustering result, we added color bars below the plots of Single Link and OPTICS, where the colors refer to the original class labels of the points in the data set. Furthermore, we chose the measures described in Section 2.2 to provide a quantitative comparison of the clustering results.

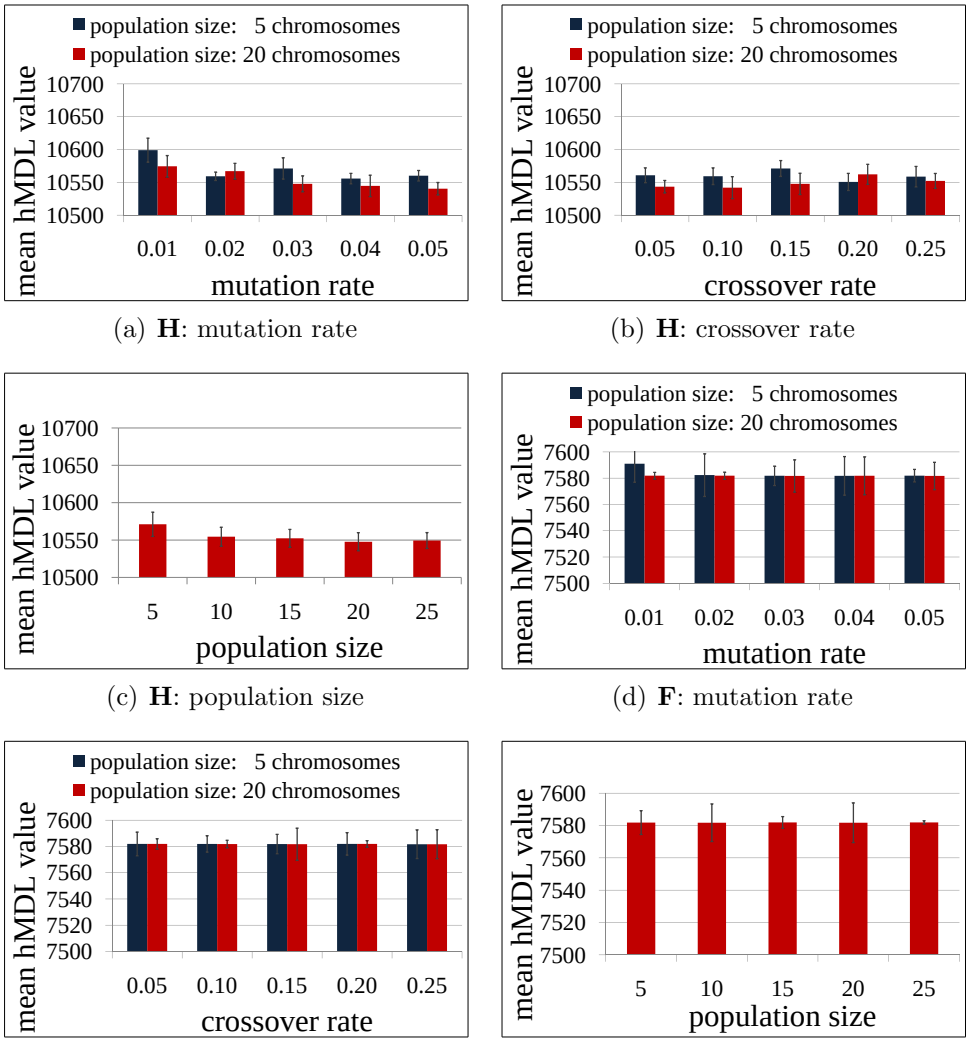
Evaluation of Genetic Parameters

We applied GACH on two different data sets to evaluate the mutation and crossover rates and the impact of the population size on the quality of the results w.r.t. the fitness function, introduced in Section 3.3.1. One data set consists of 1360 2-dimensional data points that form a true hierarchy of six clusters. The second data set covers 850 2-dimensional data points

that are grouped in two flat clusters. For each experiment, we present the mean $hMDL$ value and the corresponding standard deviation over ten runs. GACH turned out to be very robust and determines very good clustering results ($Prec > 90\%$, $Rec > 90\%$) indepent of the parametrizations.

Different Mutation Rates. We evaluated different mutation rates ranging from 1% to 5% on two different population sizes and a fixed crossover rate of 15%. As a mutation within a HCS is performed by one of the four operations `delete`, `add`, `demote` or `promote` the mutation rate is the sum of p_{del}, p_{add}, p_{dem} and p_{pro} (cf. Section 3.3.1). As `demote` and `promote` turned out to be essential for the quality of the clustering results p_{dem} and p_{pro} are typically parametrized by a multiple of p_{del} or p_{add}. This is due to the fact that the optimal number of clusters which is influenced by p_{del} and p_{add} is determined very fast by the fitness function, but p_{dem} and p_{pro} have an impact on the hierarchical structure of the clusters that has to be adjusted during the run of GACH. Figures 3.10(a) and 3.10(d) demonstrate that the mutation rate has no outstanding effect on the clustering result, neither on a hierarchical nor on a flat data set. Higher mutation rates result in higher runtimes (3388 ms for mutation rate = 0.05 vs. 1641 ms for mutation rate = 0.01 on hierarchical data set, population size = 5). However, a higher mutation rate provides more flexibility. Hence, we achieved slightly better results with a mutation rate of 0.05 ($hMDL = 10520$) compared to a mutation rate of 0.01 ($hMDL = 10542$).

Different Crossover Rates. We compared the clustering result for different crossover rates p_{co} ranging from 0.05 to 0.25 in combination with a mutation rate of 0.03 on two different population sizes. Figures 3.10(b) and 3.10(e) show that the performance of GACH is almost stable w.r.t. the different parameterizations of p_{co}. Especially on the flat data set a higher value of p_{co} has no impact on the clustering result. GACH achieved a nearly optimal



(a) **H**: mutation rate

(b) **H**: crossover rate

(c) **H**: population size

(d) **F**: mutation rate

(e) **F**: crossover rate

(f) **F**: population size

Figure 3.10: Mean fitness of resulting clusterings over ten runs on (**H**)ierarchical and (**F**)lat data sets w.r.t. the genetic parameters mutation rate, crossover rate and population size.

$hMDL$ value in almost every run, even for relatively small population sizes. Higher p_{co} values enable GACH to search the data space more effectively as the crossover between two strong individuals produces an even fitter individual. Therefore, we need less generations to find good clustering results, e.g. the result of GACH on the hierarchical data set using five structures was determined after 75 generations (1993 ms per generation) with $p_{co} = 0.05$, and after 61 generations (2553 ms per generation) with an crossover rate of $p_{co} = 0.25$.

Different Population Sizes. We tested the impact of the population size on the quality of the clustering result. We used populations that cover 5, 10, 15, 20 and 25 hierarchical cluster structures in combination with a mutation rate of 3% and a crossover rate of 15%. Figures 3.10(c) and 3.10(f) show again the mean $hMDL$ value over ten runs for each population size on two different data sets. Both plots demonstrate that a higher population size tends to produce better results, which can be explained by the fact that a higher population size provides more variation opportunities whereby a global optimum can be reached easier. However, a large number of chromosomes cause a considerable amount of runtime. One generation using 5 chromosomes took 2462 ms on average, the computation of a generation on 25 chromosomes took 9229 ms.

Hence we use a population size consisting of ten cluster structures in combination with a mutation rate of 0.03 and $p_{co} = 0.15$ in the following experiments.

Synthetic Data

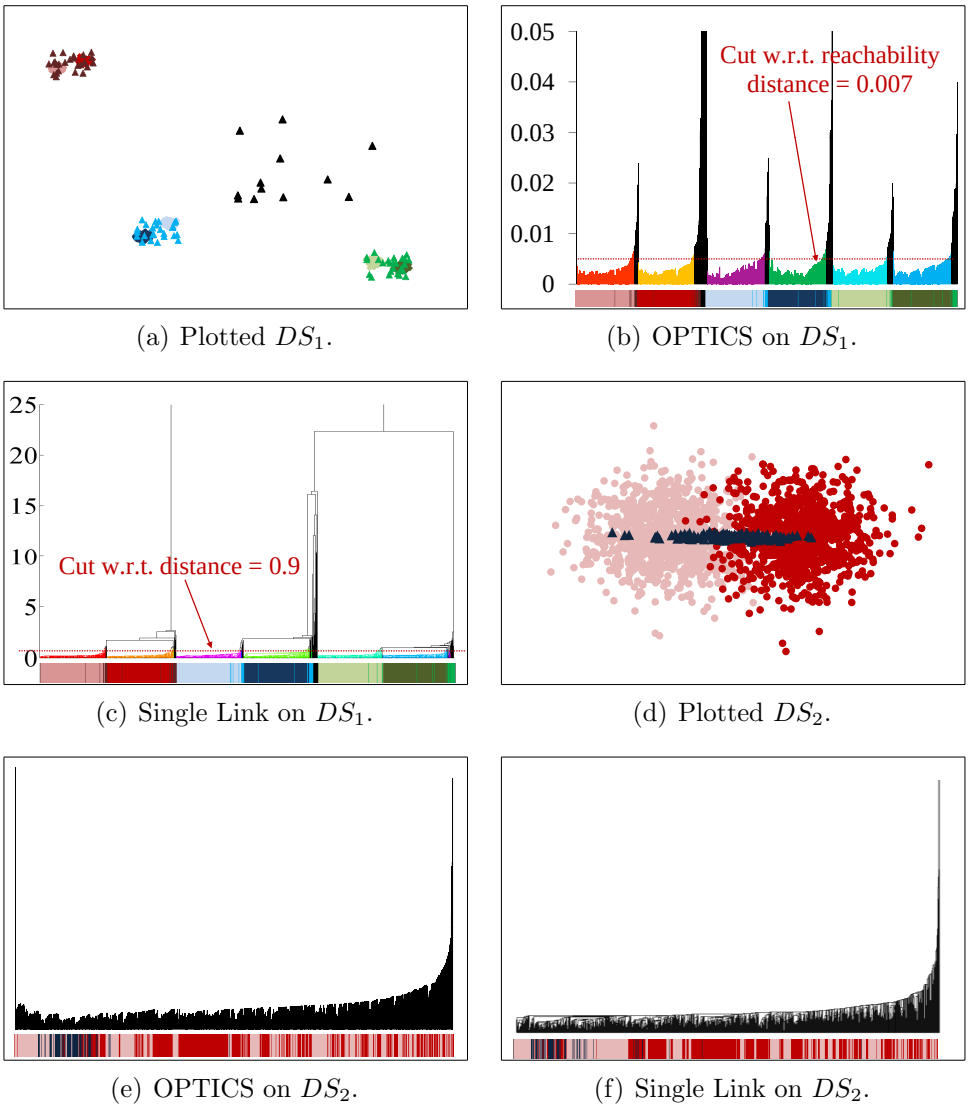
For these experiments we use two different synthetic data sets DS_1 and DS_2. DS_1 is composed of 987 2-dimensional data points that form a hierarchy of six clusters surrounded by local and global noise (cf. Figure 3.11(a)). DS_2

Table 3.5: Performance of GACH on DS_1.

	GACH	ITCH	RIC	OPTICS	Single Link
NMI	0.9346	0.9265	0.8673	0.9045	0.7429
AMI	0.9159	0.8999	0.7678	0.8662	0.5611
PREC	0.9404	0.9222	0.6438	0.8626	0.2226
REC	0.9514	0.9422	0.7720	0.9200	0.4620
DOM	0.4193	0.4454	0.6638	0.4960	1.0423

consists of 1,950 (2-dimensional data points that are grouped in three flat strongly overlapping clusters (cf. Figure 3.11(d)).

Evaluation w.r.t. Data Set $\mathbf{D_1}$. Table 3.5 presents the quantitative comparison of GACH on data set DS_1 which is plotted in Figure 3.11(a). For OPTICS, a reachability distance of 0.9, and for Single Link a cut at distance = 0.007 was used for calculating the measures. It shows that GACH outperforms all other methods concerning the quality measures given in Table 3.5. 94% of all data points are assigned to the true cluster (precision) and 95% of the cluster contents were detected correctly by GACH (recall). The reachability plot of OPTICS is given in Figure 3.11(b) which finds the correct cluster structure and separates the outliers from the cluster points. The dendrogram of Single Link 3.11(c) seems to find the correct cluster structure but performs worst when assigning the points to the correct clusters (Precision 22%, Recall 46%). However, the color bars below the plots of Single Link and OPTICS indicate that outliers are wrongly assigned to clusters. ITCH shows better results than OPTICS, RIC or Single Link but cannot measure up to the results produced by GACH. However, both GACH and ITCH were able to determine the right cluster structure but GACH outperformes ITCH w.r.t. accuracy, as GACH results in a different points to clusters assignment.



(a) Plotted DS_1.

(b) OPTICS on DS_1.

(c) Single Link on DS_1.

(d) Plotted DS_2.

(e) OPTICS on DS_2.

(f) Single Link on DS_2.

Figure 3.11: Competitive evaluation of GACH on two different synthetic data sets. DS_1 forms a hierarchy including local and global noise, DS_2 is a flat data set of three overlapping clusters.

Table 3.6: Performance of GACH on DS_2.

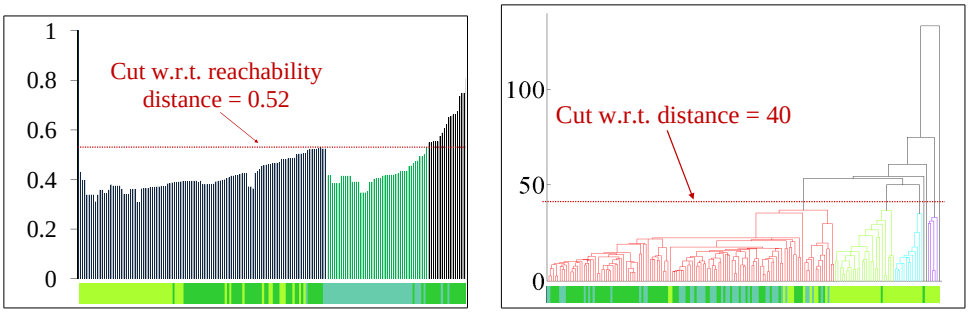
	GACH	ITCH	RIC	OPTICS	Single Link
NMI	0.6698	0.6316	0.0000	0.0029	0.0132
AMI	0.5877	0.4030	0.0000	0.0029	0.0000
PREC	0.9184	0.8227	0.2130	0.5016	0.6750
REC	0.9226	0.8913	0.4615	0.4626	0.4631
DOM	0.3325	0.4226	0.9184	0.9199	0.9224

Evaluation w.r.t. Data Set $\mathbf{D_2}$. Neither OPTICS nor Single Link were able to detect the true cluster structure of DS_2. Both fail because of a massive chaining effect and therefore the reachability plot provided by OPTICS (cf. Figure 3.11(e)) and the dendrogram produced by Single Link (cf. Figure 3.11(f)) do not uncover any cluster structure. Furthermore, the color bars below the plots indicate that the points are arranged in almost arbitrary order. RIC determines only one single cluster. ITCH separates the two red Gaussian clusters but fails in assigning the data points generated by the blue cluster correctly. Hence, GACH turned out to be the only algorithm that handles data sets with strongly overlapping clusters successfully. It shows the best values w.r.t. the quality criteria, while being very accurate. Its result causes only a DOM value of 0.3325 compared to more than 0.9 for almost all other approaches. The evaluation on DS_2 is summarized in Table 3.6. GACH achieves best result w.r.t. all quality measures.

Application of GACH on Real World Data

The *Wine* data set, available at the UCI Machine Learning repository[1] contains 178 13-dimensional data objects resulting from a chemical analysis of wines grown in the same region in Italy but derived from three different culti-

[1] `http://archive.ics.uci.edu/ml/datasets/Wine`



(a) OPTICS on wine data. (b) Single Link on wine data.

Figure 3.12: Clustering result on wine data.

vars. The data set provides a ground-truth classification that structures the data into one root node covering the whole data set and three subclusters defining the three cultivars. This structure was only determined by GACH resulting in high validity values (cf. Table 3.7). Most of the competitors did not even find the right number of clusters. With a cut at reachability distance = 0.52, OPTICS detects two clusters and some outliers (cf. Figure 3.12(a)). The color bar below the OPTICS plot indicates that only one cluster can be identified clearly. On the other hand, Single Link results in four clusters when cutting the dendrogram at distance 40. RIC even merges all data points in only one single cluster. The quantitative comparison pre-

Table 3.7: Performance on wine data.

	GACH	ITCH	RIC	OPTICS	Single Link
NMI	0.7886	0.7615	0.0000	0.5079	0.0380
AMI	0.7813	0.6912	0.0000	0.4817	0.0084
Prec	0.9401	0.9737	0.1591	0.7466	0.1564
Rec	0.9326	0.8596	0.3989	0.6966	0.3876
DOM	0.3631	0.3285	1.1405	0.6853	1.1924

sented in Table 3.7 shows that only ITCH is able to get similar results in terms of accuracy. All other methods fail to find the correct cluster structure nor the correct point to cluster assignment.

Chapter 4

Clustering Mixed Type Data

In the previous Chapter, novel algorithms for hierarchical clustering have been proposed which are based on a hierarchical variant of the Minimum Description Length (MDL) principle that interprets the hierarchical cluster structure as a statistical model of the data set. In this way, no knowledge about the data set is needed in order to identify valid and meaningful clusters in the data space. However, these methods can only deal with numerical valued attributes and cannot be applied to data sets with mixed-type valued attributes. In this Chapter, we propose INTEGRATE, a novel clustering method that integrates the information provided by heterogeneous numerical and categorical attributes. Based on the MDL principle it allows a unified view on numerical and categorical information and naturally balances the influence of both sources of information.

After a general introduction to integrative clustering in the next Section, Section 4.2 provides a survey on clustering methods for mixed-type attributes. Section 4.3 introduces in detail $iMDL$, a novel information-theoretic clustering quality criterion suitable for integrative clustering. Section 4.4 provides the details of the new INTEGRATE algorithm that is applied to several synthetic and real-world data sets (cf. Section 4.5). The basic ideas contained in this Chapter have been published in [21].

4.1 Introduction

Integrative mining of heterogeneous data is one of the grand challenges for data mining in the next decade. Recently, the need for integrative data mining techniques has been emphasized in panel discussions, e.g. at KDD 2006 [139], in workshops [181] and in position papers, [177, 98]. Integrative data mining is among the top 10 challenging problems in data mining identified in [177]. Moreover, it is essential for solving many of the other top 10 challenges, including data mining in social networks and data mining for biological and environmental problems.

In the following Sections, we focus on integrative clustering and we address the question of how to find a natural clustering of data with mixed type attributes. Not only in social network analysis and bioinformatics, data with mixed type attributes are prevalent. In everyday life, huge amounts of such data are collected, for example from credit assessments. The collected data include numerical attributes, such as credit amount and age, as well as categorical attributes, such as personal status and the purpose of the credit. A cluster analysis of credit assessment data is interesting, e.g., for target marketing. However, finding a natural clustering of such data is a non-trivial task. We identified two major problems:

- *Problem 1:* Much previous knowledge required.
- *Problem 2:* No adequate support of mixed type attributes.

Most clustering algorithms require previous knowledge on the data in the form of input parameters which are difficult to estimate, for example the number of desired clusters k. Recently, great efforts have been made in developing clustering methods that avoid such difficult parameter settings, e.g. [18, 19, 20, 133]. However, as most approaches to clustering, these algorithms are designed for clustering vector data and provide no support for mixed type

attributes. Recently, some approaches for integrative clustering have been proposed (cf. Section 4.2). However, the parametrization of most of these algorithms is very difficult. Besides the parameters required for clustering itself, mixed type attributes introduce additional parameters specifying the relative importance of numerical and categorical attributes in the clustering process.

To cope with these two major problems, we propose INTEGRATE, a parameter-free technique for integrative clustering of data with mixed type attributes. The major benefits can be summarized as follows:

1. Natural balance of numerical and categorical information in clustering supported by information theory.
2. Parameter-free clustering.
3. Making most effective usage of numerical as well as categorical information.
4. Scalability to large data sets.

INTEGRATE is based on iMDL, an information-theoretic clustering quality criterion suitable for integrative clustering. Regarding clustering as a data compression problem, the Minimum Description Length (MDL) principle allows a unified view on numerical and categorical information. This unified view naturally balances the influence of numerical and categorical attributes in clustering without requiring any weighting parameters.

4.2 Related Work

One of the first approaches to integrative clustering is k-prototypes[84] which combines k-means for clustering numerical data with k-modes for categorical

data for clustering data with mixed type attributes. This algorithm uses a weighted sum of Euclidean distance for numeric attributes and the simple matching dissimilarity measure for categorical attributes. However, the attribute weights and the number of clusters have to be determined a priori.

CFIKP [178] can process large data sets using a CF*-tree to pre-cluster the data set into dense regions which are then clustered by an improved k-prototypes algorithm. Hence, the problem for selecting the number of clusters still remains. Also the system by Hsu *et al.* [83] suffers from this drawback of difficult parameter settings. CAVE is an incremental entropy-based method which first selects k clusters and then assigns objects to these clusters based on variance and entropy. The method combines variance for measuring the similarity of numeric values and integrates entropy with distance hierarchies for categorical attributes. Knowledge of the similarity among categorical attributes is needed in order to construct the distance hierarchy for the categorical attributes.

The authors in [78] present the cluster ensemble approach CEBMDC that overcomes the problem of selecting k but requires as input parameter a threshold that defines the intra-cluster similarity between objects. In this framework, a divide-and-conquer technique is applied to solve the clustering problem separately for numerical and categorical data respectively, and to combine the results to get the final clusters.

In [147] the authors introduce the CBC algorithm for clustering mixed data type based on compressed data that is an extension of the BIRCH [179] algorithm. This method uses a weight-balanced tree that needs two parameters: B and L. B defines the number of entries for non-leaf nodes whereas L gives the number of entries for leaf nodes. Furthermore all entries in a leaf node must satisfy a threshold requirement, w.r.t. a similarity value T. The

method has good scalability but requires difficult parameter settings.

Ahmad and Dey [2] propose a k-means-based method for mixed numeric and categorical attributes, but the process of solving the optimization of the cost function is very complex and it is not efficient for large data sets.

The authors in [26] use standard fuzzy c-means on a set of feature vectors with mixed features which was mapped to a set of feature vectors with only real valued components. This new set of vectors with only numeric components, is then clustered using the traditional fuzzy clustering method [15]. However, this mapping is computationally very intensive and is designed rather for very low dimensional data. Moreover, the method uses an error function that may have a large number of local optima.

A recently proposed study [107] by Li and Chen is the IWEKM algorithm which extends the cost function of entropy weighting k-means clustering algorithm [88] to more effectively specify the inter and intra-cluster similarities. This is done by adding a variable that is linear to the square sum of the distances from the mean of all objects to the mean of all clusters and therefore minimizes the distances between objects inside each cluster and maximizes the inter cluster distances. Another variable is added that quotes the degree similarity between two objects considering the categorical attributes. For this method many input parameters have to be specified in advance which cannot be chosen intuitively.

Many existing clustering methods for mixed type data are based on k-means [111] and k-modes [84] which are most widely used methods for clustering continuous or categorical data, respectively. A major drawback of these methods is the difficult choice of the parameter k which defines the number of clusters a priori.

4.3 Minimum Description Length for Integrative Clustering

Notations. In the following we consider a data set DS with n objects. Each object x is represented by d attributes. Attributes are denoted by capital letters and can be either numerical features or categorical variables with two or more values. For a categorical attribute A, we denote a possible value of A by a. The result of our algorithm is a disjoint partitioning of DS into k clusters $\mathcal{C}_1, ..., \mathcal{C}_k$.

Likelihood and Data Compression. One of the most challenging problems in clustering data with mixed attribute type is selecting a suitable distance function, or unifying clustering results obtained on the different representations of the data. Often, the weighting between the different attribute types needs to be specified by parameter settings, cf. Section 4.2. The minimum description length (MDL) principle provides an attractive theoretical foundation for parameter-free integrative clustering avoiding this problem. Recently, the MDL-principle and related ideas have been successfully applied to avoid crucial parameter settings in clustering vector data, but to the best of our knowledge have not been applied to integrative clustering so far. Regarding clustering as a data compression problem allows us a unifying view, naturally balancing the influence of categorical and numerical attributes. Probably the most important idea of MDL which allows integrative clustering is relating the concepts of likelihood and data compression. Imagine data needs to be transferred via a communication channel from a sender to a receiver. If the data exhibits some regularities or patterns, the communication cost can be minimized by applying an appropriate coding scheme exploiting these patterns. Data compression can be maximized by assigning short descriptions to regular data objects which exhibit the characteristic patterns and longer descriptions to the few irregular objects.

Coding Categorical Data. To give a simple example, assume we need to transfer 1,000 one-dimensional categorical data objects. Each object is represented by a categorical attribute A with two possible values *red* and *blue*. It can be shown that the code length to transfer this data is lower bounded by the entropy of the attribute A. Thus, the coding cost cc of attribute A is provided by:

$$CC(A) = -\sum_{a \in A} p(a) \cdot \log_2 p(a).$$

Here, a denotes each possible value or outcome of the categorical attribute. By the application of the binary logarithm we obtain the code length in bits. If we have no additional knowledge on the data we have to assume that the probabilities for *red* and *blue* are both 0.5. With this assumption, we need one bit to encode each data object and the communication cost would be 1,000 bits. Clustering, however, provides high-level knowledge on the data which allows for a much more effective way to reduce the communication cost. Even if the probabilities for the different outcomes of the categorical attributes are approximately equal considering the whole data set, often different clusters with non-uniform probabilities can be found.

As an example, refer to the data displayed in Figure 4.1. The data are represented by two numerical attributes (which we ignore for the moment) and one categorical attribute. The categorical attribute has two possible values, *red* and *blue*, which are visualized by the corresponding colors. Considering all data, the probabilities for *red* and *blue* are both 0.5. However, it is evident that the outcomes *red* and *blue* are not uniformly distributed in all areas of the data space. Rather, we have two clusters, one preliminarily hosts the red objects, and the other the blue ones. In fact, the data has been generated such that in the cluster displayed on the left, we have 88% of blue objects and 12% of red objects. For the cluster on the right, the ratio has

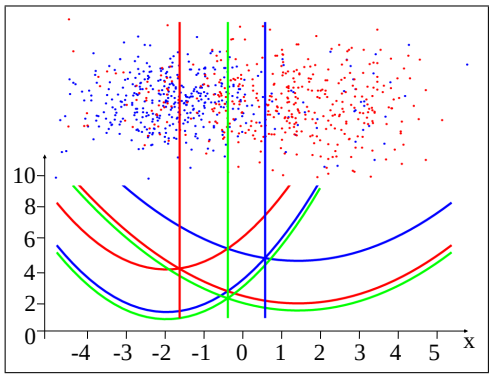


Figure 4.1: Top: Example data set with two numerical and one categorical attribute with the outcomes *red* and *blue*. Bottom: Cost curves assuming two clusters: Considering the numerical information only (green), integrating numerical and categorical information (red, blue). For each outcome and each cluster, we have a unique cost curve. Intersection points mark the resulting cluster borders.

been selected reciprocally. This clustering drastically reduces the entropy and therefore the coding cost of the categorical attribute to $CC(A) = 0.53$ bits per data object, which corresponds to the entropy of A in both clusters. However, we would need to transfer the clustering result itself, including the cluster identifiers to the receiver. Before discussing how to encode the clustering result, we will consider how to encode numerical data.

Coding Numerical Data. If our data set contains an additional numerical attribute B we can also use the relationship between likelihood and data compressibility to reduce the communication cost. To specify the probability of each data object considering attribute B, we assign a probability density function (PDF) to B. We apply a Gaussian PDF for each numerical attribute. However, let us note that our ideas can be straightforwardly extended to other types PDF, e.g. Laplacian or Generalized Gaussian. Thus, the probability

density function of a numerical attribute B is provided by:

$$p(b) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(b-\mu_B)^2}{2\sigma_B^2}\right).$$

If the data distribution of attribute B is Gaussian with mean μ_B and standard deviation σ_B, we minimize the communication cost of the data by a coding scheme which assigns short bit strings to objects with coordinate values that are in the area of high probability and longer bit strings to objects with lower probability. Let us note that we do not consider how to actually construct such code. Rather, we are interested in the code length as a measure for clustering quality. The coding cost of attribute B is provided by:

$$CC(B) = -\int p(b) \log_2\ p(b)\mathbf{d}b.$$

Again, if we have no knowledge on the data, we would have to assume that each attribute is represented by a single Gaussian with mean and standard deviation determined from all data objects. As discussed for categorical data, clustering can often drastically reduce the communication cost. Most importantly, relating clustering to data compression allows us a unified view on data with mixed type attributes.

Consider again the data displayed in Figure 4.1. In addition to the categorical attribute we now consider the numerical x-coordinate, denoted by X. To facilitate presentation, we ignore the y-coordinate which is processed analogously. The two green curves at the bottom represent the coding costs of the two clusters considering X. For both curves, the cost minimum coincides with the mean of the Gaussian which generated the data. The cluster on the right has been generated with slightly larger variance, resulting in slightly higher coding costs. The intersection of both cost curves represents the border between the two clusters provided by X, indicated by a green vertical line. In addition, for each cluster and each outcome of the categorical attribute, we have included a cost curve (displayed in the corresponding

colors). Again, the intersection points mark the cluster borders provided by the categorical attribute. Consider, e.g., the red vertical line. Red objects with a value in X beyond that point are assigned to the cluster on the right. Thus, in the area between the red and the blue vertical line, the categorical value is the key information for clustering. Note that all borders are not fixed but optimized during the run of our algorithm.

A Coding Scheme for Integrative Clustering. As mentioned, in addition to the coding cost for categorical and numerical attributes of the data objects, we need to elaborate a coding scheme describing the clustering result itself. The additional coding cost for encoding the clustering result can be classified into two categories: the *parameter cost* required to specify the cluster model and the *id-cost* required to specify the cluster-id for each object, i.e. the information to which cluster the object belongs.

For the parameter cost, let us focus on the set of objects belonging to a single cluster $\mathcal{C}$. To specify the cluster model, we need for each categorical attribute A to encode the probability of each value or outcome a. For a categorical attribute with $|A|$ possible values, we need to encode $|A| - 1$ probabilities since the remaining probability is implicitly specified. For each numerical attribute B we need to encode the parameters μ_B and σ_B of the PDF. Following a central result from the theory of MDL [148], the parameter cost to model the $|\mathcal{C}|$ objects of the cluster can be approximated by $p/2 \cdot \log_2 |\mathcal{C}|$, where p denotes the number of parameters. The parameter cost depends logarithmically on the number of objects in the cluster. The considerations behind this are that for clusters with few objects, the parameters do not need to be coded with very high precision. To summarize, the parameter cost for a cluster $\mathcal{C}$ are provided by

$$PC(\mathcal{C}) = \frac{1}{2} \cdot ((\sum_{A_{cat}} |A| - 1) + |B_{num}| \cdot 2)) \cdot \log_2 |\mathcal{C}|.$$

Here A_{cat} stands for all categorical attributes and B_{num} for all numerical attributes in the data. Besides the parameter cost, we need for each object to encode the information to which of the k clusters it belongs. Also for the id-cost, we apply the principle of Huffman coding which implies that we assign shorter bit-strings to the larger clusters. Thus, the id-cost IDC of a cluster $\mathcal{C}$ are provided by:

$$IDC(\mathcal{C}) = \log_2 \frac{n}{|\mathcal{C}|}.$$

Putting it all together, we are now ready to define $iMDL$, our information-theoretic optimization goal for integrative clustering.

$$iMDL = \sum_{\mathcal{C}} (\sum_{Attr} |\mathcal{C}| \cdot CC(A)) + PC(\mathcal{C}) + IDC(\mathcal{C}).$$

For all clusters $\mathcal{C}$ we sum up the coding cost for all numerical and categorical attributes $Attr$. To the coding cost we need to add the parameter cost and the id-cost of the cluster, denoted by $PC(\mathcal{C})$ and $IDC(\mathcal{C})$, respectively. Finally, we sum up these three terms, coding cost, parameter cost and id-cost for all k clusters.

4.4 Algorithm INTEGRATE

INTEGRATE is designed to find the optimal clustering of a data set DS, where each object x comprises both numerical and categorical attributes by optimizing the overall compression rate. INTEGRATE is built on top of a k-means algorithm. Therefore, a straightforward approach would be to run INTEGRATE with a predefined number of clusters k in order to get a partition of the data set into k clusters. The pseudocode for this baseline approach is given in Algorithm 2.

Algorithm 2 INTEGRATE(DS, k)

```
Initial clustering of DS with k initial clusters C_1, ..., C_k;
change = true;
while change do
  change = false;
  for all x ∈ DS do
    mincost ← cost(C_1, ..., C_k);
    Assign x to cluster C_1;
    for all C ∈ {C_1, ..., C_k} do
      newcost ← cost(C)
      if newcost < mincost then
        mincost ← newcost;
        Assign x to cluster C;
      end if
    end for
    if the cluster assignment of x changes then
      change ← true;
    end if
    for all C ∈ {C_1, ..., C_k} do
      Recalculate the parameters of C according all assigned data objects;
    end for
  end for
end while
```

Ensure: Optimal clustering $C_1, \ldots, C_k$;

First, INTEGRATE builds an initial partitioning of k clusters. Each cluster is represented by a Gaussian PDF in each numerical dimension B with μ_B and σ_B, and a probability for each value of the categorical attributes. All objects are then assigned to the k clusters by minimizing the overall coding cost $iMDL$. In the next step, the parameters of each cluster C are recalculated according to the assigned objects. That implies the μ and the σ in each numerical dimension B and the probabilities for each value of the categorical attributes, respectively.

The μ and the σ for a numerical Attribute B is given by:

$$\mu_{C_B} = \frac{\sum\limits_{x \in C} x_{i,B}}{|C|}, \qquad \sigma_{C_B} = \sqrt{\frac{\sum\limits_{x \in C} (x_{i,B} - \mu_{C_B})^2}{|C|}}$$

The probability for the value a of a categorical attribute A is determined in the following way:

$$p(a) = \frac{\sum\limits_{x \in C} \begin{cases} 1 & a_d = a \\ 0 & else \end{cases}}{|C|}.$$

After initialization the following steps are performed repeatedly until convergence.
First, the cost for coding the actual cluster partition are determined. Second, assignment of objects to clusters is performed in order to decrease the $iMDL$ value. Third, the new parameters of each cluster are recalculated. The algorithm terminates if no further changes of point to cluster assignments occur. Finally, INTEGRATE returns the optimal clustering for data set DS, represented by k clusters according to minimum coding costs.

The effectiveness of an algorithm often heavily depends on the quality of the initialization, as it is often the case that the algorithm can get stuck in a local optimum. Hence, we propose an initialization scheme technique to avoid this effect.

Initialization of INTEGRATE. We have to find initial cluster representatives that correspond best to the final representatives. An established method for partitioning methods is to initialize with randomly chosen objects of the data set. We adopt this idea and take the μ of the numerical attributes of k randomly chosen objects as cluster representatives. During initialization, we set $\sigma = 1.0$ in each numerical dimension. The probabilities of the values for the categorical attributes are set to $\frac{1}{|a|}$. Then a random set of $\frac{1}{z}n$ objects is selected, where n is the number of objects in data set DS and $z = 10$

turned out to give satisfying results. Finally, we chose the clustering result that minimizes the coding cost best, within m initialization runs. Typically $m = 100$ runs suffice for an effective result.

Automatically Selecting the Number of Clusters k. The effectiveness of INTEGRATE can further be improved using $iDML$ as optimization criterion to avoid difficult parameters like the number of clusters k. Using the *iMDL* criterion for mixed type data, we can avoid the parameter k. As an optimal clustering that represents the underlying data structure best has minimum coding costs, $iMDL$ can also be used to detect the number of clusters. For this purpose, INTEGRATE uses $iMDL$ no longer exclusively as selection criterion for finding the correct object to cluster assignment. Rather we now estimate the coding costs for each k where k is selected in a range of $1 \leq k \leq n$. For efficiency reasons INTEGRATE performs this iteration step on a $z\%$ sample of the data set. The global minimum of this cost function gives the optimal k and thus the optimal number of clusters.

4.5 Experiments

This Section provides a detailed experimental evaluation of INTEGRATE. Besides synthetic data, several real world data sets have been used. Since INTEGRATE is a hybrid approach combining the benefits of clustering methods using only numeric attributes and those for categorical attributes we compare algorithms of both categories and algorithms that can also handle mixed type attributes. In particular, we selected the k-means-based method by Ahmad and Dey [2] denoted by KMM and the algorithm k-prototypes [84]. Furthermore, we compare to the k-means algorithm [111] which is probably the most common approach to clustering numerical data. As a representative for clustering categorical data we selected the widely used method k-modes [84]. For k-means and k-modes the numerical and categorical attributes were ignored.

For evaluation, we used the validity measure by [49] referred to as DOM in the following. For details on clustering validation techniques, please refer to Section 2.2. We report in each experiment the average performance of all clustering algorithms over 10 runs.

4.5.1 Synthetic Data

If not otherwise specified the artificial data sets include three Gaussian clusters with each object having two numerical attributes and one categorical attribute. To validate the results we added a class label to each object which was not used for clustering.

Varying Ratio of Categorical Attribute Values. In this experiment, we generated three-dimensional synthetic data sets with 1,500 points including two numerical and one two-valued categorical attribute. We varied the ratio for each of the values of the categoric attributes from 1:0 to 0:1 clusterwise in each data set. Without need for difficult parameter setting our proposed method performs best in all cases (cf. Figure 4.2(a)). Even in the case of equally (5:5) distributed values, where the categorical attribute gives no information for separating the objects, the cluster quality of INTEGRATE is best compared to all other methods. As k-means does not take the categorical attributes into account the performance is relatively constant.

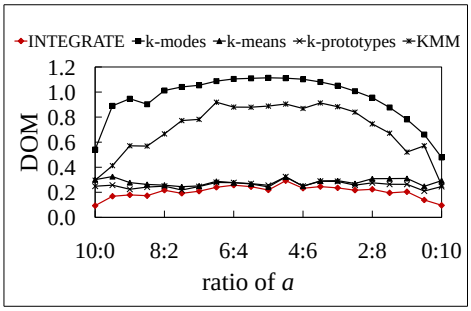
Varying Variance of Clusters. This experiment aims at comparing the performance of the different methods on data sets with varying variances. In particular, we generated synthetic data sets each comprising 1,500 points including two numerical and one two-valued categorical attribute that form three Gaussian clusters with a variance ranging from 0.5 to 2.0. Figure 4.2(b) shows that INTEGRATE outperforms all competitors in all cases, in which each case reflects different degree of overlap of the three clusters. Even at a variance of 2.0 where the numerical attributes carry nearly no cluster in-

formation, our proposed method shows best cluster quality as in this case the categorical attributes are used to separate the clusters. On the contrary, k-modes performs worst as it can only use the categorical attribute as single source for clustering.

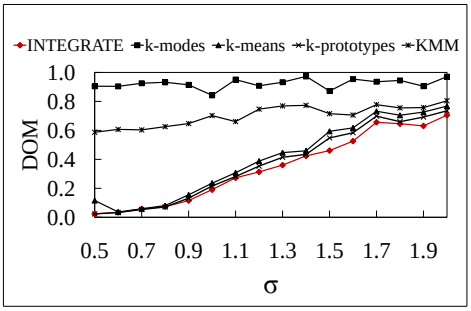
Varying Clustersize. In order to test the performance of the different methods on data sets with unbalanced clustersize we generated three Gaussian clusters with different variance and varied the ratio of number of points per cluster from 1:10:1 to 10:1:10 in five steps. It is obvious from Figure 4.2(c) that INTEGRATE best separates the three clusters even with highly unbalanced clustersizes. Only in the case of two very small clusters and one big cluster (1:10:1) k-modes shows a slightly better cluster validity.

Varying Number of Numerical Dimensions. In this experiment, we leave the number of categorical attributes to a constant value and successively add numerical dimensions to each object that are generated with a variance of σ=1.8. INTEGRATE shows best performance in all cases (cf. Figure 4.2(d)). All methods show a slight increase in cluster quality when varying the numerical dimensionality, except k-modes that performs constantly as it does not consider the numerical attributes.

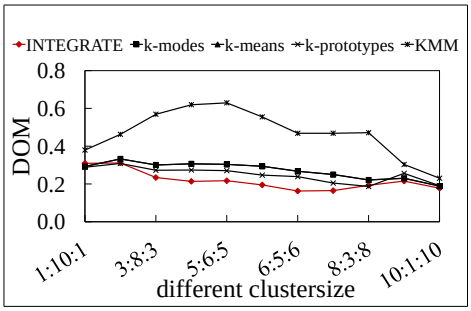
Varying Number of Categorical Dimensions. For each object, we added three-valued categorical attributes where we set the probablity of the first value to 0.6 and the probability of the two remaining values to 0.2, respectively. Figure 4.2(e) illustrates that our proposed method outperforms the other methods and even k-modes by magnitudes which is a well-known method for clustering categorical data. Whereas KMM shows a heavy decrease in clustering quality in the case of two and four additional categorical attributes, our method performs relatively constant. Taking the numerical attributes not into account the cluster validity of k-means remains constant.



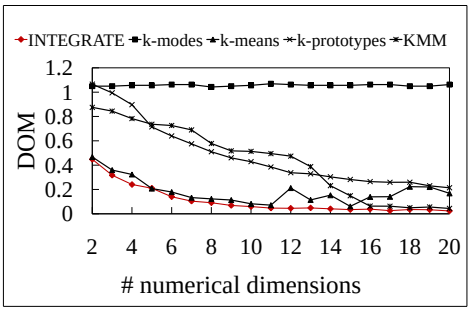
(a) Dom value vs. different ratio of attribute values.



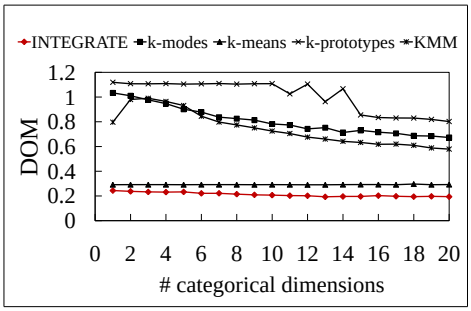
(b) Dom value vs. different variance of the clusters.



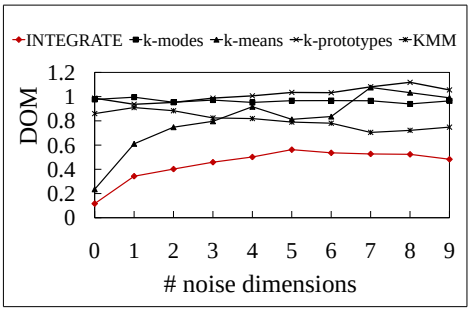
(c) Dom value vs. different clustersizes.



(d) Dom value vs. different number of numerical dimensions.



(e) Dom value vs. different number of categorical dimensions.



(f) Dom value vs. number of noise dimensions.

Figure 4.2: Synthetic data

Noise Dimensions. Figure 4.2(f) illustrates the performance of the different methods on noisy data. It it obvious that INTEGRATE outperforms all compared methods when adding non-clustered noise dimensions to the data. k-means shows a highly increase in the DOM values which refers to decreasing cluster validity. Even in the case of nine noise dimensions INTEGRATE leads to the best clustering result.

4.5.2 Real Data

Finally, we show the practical application of INTEGRATE on real world data, available at the UCI repository [9]. We chose two different data sets with mixed numerical and categorical attributes. An additional class attribute allows for an evaluation of the results. Table 4.1 reports the μ and σ of the DOM value for all methods within 10 runs. For all compared methods we set k to the number of classes.

Heart Disease. The heart disease data set comprises 303 instances with six numerical and eight categorical attributes each labeled to an integer value between 0 and 4 which refers to the presence of heart disease. Without any prior knowledge on the data set we obtained best cluster validity of 1.23 with INTEGRATE. KMM performed slightly worse. However, the runtime of INTEGRATE is 0.1 seconds compared to KMM which took 2.8 seconds to return the result.

Credit Approval. The Credit Approval data set contains results of credit card applications. It has 690 instances, each being described by six numerical and nine categorical attributes and classified to the two classes 'yes' or 'no'. With a mean DOM value of 0.61 INTEGRATE best separated the objects into two clusters. It took 0.1 seconds to return the result. Note, that INTEGRATE requires no input parameter in order to find the optimal clustering

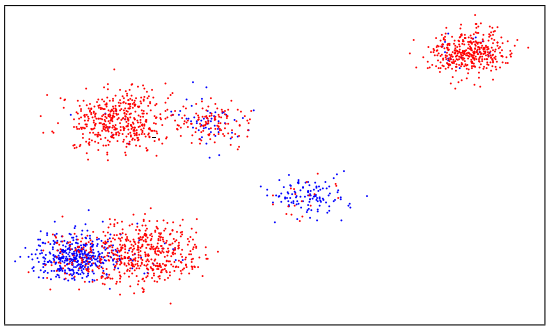
Table 4.1: Results of INTEGRATE on real data.

		INTEGRATE	k-means	k-modes	kMM	k-prototypes
Heart Disease	μ	1.23	1.33	1.26	1.24	1.33
	σ	0.02	0.01	0.03	0.02	0.00
Credit Approval	μ	0.61	0.66	0.70	0.63	0.66
	σ	0.03	0.00	0.00	0.09	0.00

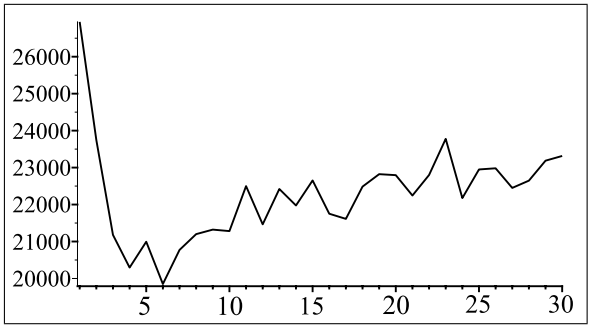
of the data. This advantage is more explicitly analyzed in the next paragraph.

4.5.3 Finding the Optimal k.

On the basis of the data set illustrated in Figure 4.3(a) we want to highlight the benefit of INTEGRATE for finding the correct number of clusters that are present in the data set. The data set comprises six Gaussian clusters with each object having two numerical attributes and one categorical attribute with two different values that are marked in red and blue, respectively. Figure 4.3(b) shows the $iMDL$ of the data model for different values of k. The cost function has its global minimum, which refers to the optimal number of clusters, at $k = 6$. In the range of $1 \leq k \leq 4$ the plotted function shows an intense decrease in the coding costs and for $k > 6$ a slight increase of the coding costs as in these cases the data does not optimally fit into the data model and therefore causes high coding costs. Note, that there is a local minimum at $k = 4$ which would also refer to a meaningful number of clusters.



(a) The data set.



(b) $iMDL$ for $1 \leq k \leq 30$.

Figure 4.3: Coding cost for different k according to a data set that consists of $k = 6$ clusters.

Chapter 5

Clustering Skylines

In the previous Chapters, novel algorithms for clustering have been proposed. All of them use a similarity measure for comparing single data points. In this Chapter, we go one step ahead and define a more complex similarity measure in order to compare different data sets that are represented by their skylines. The skyline of a data set is a well-established database primitive for finding objects in a data set which minimize two or more given attributes with an unknown weighting between theses different attributes. In order to use multiple skylines themselves as objects for data exploration and data mining we defined SkyDist, a similarity measure for comparing different skylines. In this way, SkyDist can be used for complex analysis tasks such as clustering, classification, outlier detection, etc.

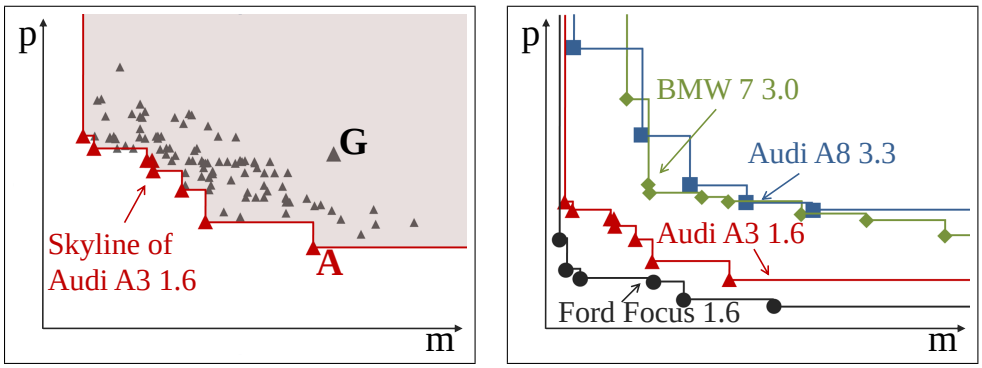
After an introduction in the next Section, Section 5.2 surveys related work. Section 5.3 introduces basic definitions necessary for the main concepts of SkyDist. In Section 5.4 we present two different algorithms for computing SkyDist, based on Monte-Carlo sampling and on the plane sweep paradigm. The Chapter ends with an experimental evaluation in Section 5.5. The concepts described in this Chapter have been published in [22].

5.1 Introduction

Skyline queries are an important area of current database research, and have gained increasing interest in recent years [23, 97, 108, 131, 160]. Most papers focus on efficient algorithms for the construction of a *single* skyline which is the answer of a user's query. In this Section, we extend the idea of skylines in such a way that multiple skylines are treated as objects for data exploration and data mining.

One of the most prominent applications of the skyline operator is to support complex decisions. As an example, consider an online marketplace for used cars, where the user wants to find out offers which optimize more than one property of the car such as p (price) and m (mileage), with an unknown weighting of the single conditions. The result of such a query has to contain all offers which may be of interest: not only the cheapest offer and that with lowest mileage but also all offers providing an outstanding combination of p and m. This concept is illustrated in Figure 5.1(a) which displays a set of database objects representing all car offers for an Audi A3 1.6 described by the attributes p and m. However, not all of these offers are equally attractive to the user. For instance, offer A has both a lower price *and* a lower mileage than G and many other offers. Therefore, we say that A *dominates* G (in symbols $A \prec G$), because A is better than G w.r.t. any possible weighting of the criteria price and mileage. More generally, the *skyline* contains all objects which are not dominated by any other objects in the data set. That is, the skyline contains all objects that exhibiting outstanding combinations of the attributes. All skyline points of the Audi A3 are highlighted in red.

For the used car market, the skyline of each car model has a particular meaning: Many arbitrarily bad offers may be present in the database but only the offers in (or close to) the skyline have a high potential to find a



(a) Database objects and their skyline. (b) Skylines of different car models.

Figure 5.1: Motivating examples for performing data mining on skylines.

customer. The skyline of the offers marks to some degree the fair value of a car for each mileage in the market. Therefore, the skyline characterizes a car model (or higher-order objects of other applications) in a highly expressive way.

This high expressiveness leads us to the idea of treating the skylines themselves as a measure of similarity. In this Chapter, we propose SkyDist, a similarity measure for skylines. SkyDist is defined by the area between two skylines. To determine SkyDist we propose two different algorithms, based on Monte-Carlo sampling and on the plane sweep paradigm. Applied in a recommender system, SkyDist can be used to point out the best alternatives having an outstanding combination of two or more attributes defined by the user. As an example, Figure 5.1(b) illustrates the skylines of four different car models derived from real data of an online automotive market. Car models which exhibit similar skylines (like Audi A3 1.6 and Ford Focus 1.6) may be considered as similar. A recommender system might find out that the Focus is a perfect alternative to the Audi A3.

However, SkyDist can also be used for more complex analysis tasks such as clustering, classification or outlier detection.

5.2 Related Work

In this Section, we briefly discuss related work. It surveys studies on skyline computation and briefly introduces clustering approaches applied to demonstrate the potential of data mining on skylines for knowledge discovery in Section 5.5. A fundamental introduction to the problem of clustering is provided in Chapter 2.1.

5.2.1 Skyline Computation

The classical skyline problem has been studied extensively in recent years, and many applications for skyline analysis have been designed. Börzsönyi *et al.* [23] proposed a SQL syntax for skyline queries and developed the Block-Nested-Loops (BNL) algorithm and the Extended Divide-Conquer algorithm. The Sort-Filter-Skyline algorithm by Chomicki *et al.* [34] improves BNL by presorting the entire data set. Tan *et al.* [160] presented two progressive algorithms, Bitmap and Index, to compute the skyline of a set of points without scanning the whole data set. Another progressive algorithm is presented by Kossmann *et al.* [97], which is based on a nearest neighbor search. Papadias *et al.* [131] developed a progressive algorithm branch-and-bound skyline, based on a nearest neighbor search technique, by browsing the data set indexed by an R-tree. Lin *et al.* [108] developed an efficient skyline computation method by determining the skyline for the most recent N elements in a data stream. All the methods mentioned here are designed for the computation of the skyline.

5.2.2 Sweep-Line Methods

The self-intersection problem of polygons is one of the main problems in computational geometry, that can be handled by two different approaches, namely line segments intersection [68, 138] and sweep-line methods [14]. Sweep-line methods are more efficient than line-segments intersections. Park

et al. [132] presented a sweep-line algorithm for finding all intersections among polygonal chains with an $O((n+k) \cdot log\ m)$ worst-case time complexity, where n is the number of line segments in the polygonal chains, k is the number of intersections, and m is the number of monotone chains. Their proposed algorithm is based on the sweep line algorithm by Bentley and Ottmann [14], which finds all intersections among a collection of line segments with an $O((n+k) \cdot log\ n)$ time complexity.

5.2.3 Clustering

In iterative partitioning clustering k-medoid methods such as PAM [94] or CLARANS [126] aim at finding a set of k representatives among all objects that characterize the objects in the data set best. Clusters are created by assigning each object to the cluster of the medoid that is closest to that object. One of the most wide spread approaches to hierarchical clustering is the Single Link algorithm [153]. Starting with singleton clusters for each object, the algorithm merges in each step the closest pair of clusters until it ends up with the root which is one large cluster containing all objects. The hierarchy obtained by the merging order is visualized as a tree which is called dendrogram. In density-based clustering, clusters are regarded as areas of high object density which are separated by areas of lower object density. The algorithm DBSCAN [55] formalizes this idea by two parameters: $MinPts$ specifying a number of objects and ϵ specifying a volume. An object is called core object if it has at least $MinPts$ objects within its ϵ-neighborhood. DBSCAN determines a non-hierarchical, disjoint partitioning of the data set into clusters.

For a more detailed description of the clustering algorithms, please refer to Chapter 2.1.

5.3 Theoretical Background

We are given a set of objects in the database, each of which is already associated with an individual skyline which has been generated from the underlying application. For instance, if our database objects are different car types, each car type is associated with the skyline of the offers that have been made in our used car market.

The objective of this Section is to define a useful distance measure for a pair of database objects which are represented by these skylines. The distance function should be useful in the sense that the characteristic properties of the skyline concept are suitably reflected in the distance measure. Whenever two skylines are similar in an intuitive sense, then the corresponding distance measure should yield a small value. In order to define such a reasonable distance measure, we recall here the central concept of the classical skyline operator, the *dominance relation* which can be built on the preferences on attributes $\mathcal{D}_1, \ldots, \mathcal{D}_d$ in a d-dimensional numeric space $\mathcal{D}$.

Definition 4 (Dominance) *For two data points u and v, u is said to* ***dominate*** *v, denoted by $u \prec v$, if the following two conditions hold:*
$\forall$ *dimensions* $\mathcal{D}_{i\in\{1,\ldots,d\}}$*:* $u.\mathcal{D}_i \leq v.\mathcal{D}_i$
$\exists$ *dimension* $\mathcal{D}_{j\in\{1,\ldots,d\}}$*:* $u.\mathcal{D}_j < v.\mathcal{D}_j$*.*

Definition 5 (Skyline Point / Skyline) *Given a set of data points DS, a data point u is a* ***skyline point*** *if there exists no other data point $v \in DS$ such that v dominates u. The* ***skyline*** *on DS is the set of all skyline points.*

Two skylines are obviously *equal* when they consist of identical points. Intuitively, one can say that two skylines are similar, whenever they consist of similar points. But in addition, two skylines may also be considered similar if they dominate approximately the same points in the data space. This can be grasped in a simple and efficient way by requiring that the one of the two skylines should not change much whenever the points of the other skyline are

inserted into the first one and vice versa. This leads us to the idea to base SkyDist on the set-theoretic difference among the parts of the data space which are dominated by the two skylines. For a more formal view let us define the terms *dominance region* and *non-dominance region* of a skyline.

Definition 6 (Dominance Region of a Skyline Point) *Given a skyline point x_i of a skyline $X = \{x_1, \ldots, x_n\}$. The **dominance region of** x_i, denoted by $\mathcal{DOM}_{x_i}$, is the data space, where every data point $u \in \mathcal{DOM}_{x_i}$ complies the condition $x_i \prec u$.*

Definition 7 (Dominance Region of a Skyline) *Given the set of all skyline points of a skyline X. The **dominance region of the skyline** X, denoted by $\mathcal{DOM}_X$, is defined over the union of the dominance regions of all skyline points $x_{i \in \{1,\ldots,n\}}$.*

Figure 5.2 illustrates this notion for two given skylines X and Y. The green and blue areas show the dominance regions of skylines X and Y, respectively.

Definition 8 (Non-Dominance Region of a Skyline) *Given a numeric space $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_d)$ and the dominance region $\mathcal{DOM}_X$ of a skyline X. The **non-dominance region of** X, denoted by $\overline{\mathcal{DOM}_X}$, is $\mathcal{D} \setminus \mathcal{DOM}_X$.*

The basic idea behind SkyDist is to determine the data space that is represented by all possible data points that are located in the dominance region of one skyline and, at the same time, the non-dominance region of the other skyline. More formally we can say that SkyDist is the volume of the distance area between two skylines which can be specified by the following equation.

$$SkyDist(X, Y) = \mathrm{Vol}((\mathcal{DOM}_X \setminus \mathcal{DOM}_Y) \cup (\mathcal{DOM}_Y \setminus \mathcal{DOM}_X)) \quad (5.1)$$

In order to determine the value of the distance of two skylines X and Y based on the concept described above, we have to limit the corresponding regions in each dimension. Therefore we introduce the notion of a bounding skyline point.

Definition 9 (Bounding Skyline Point) *Given a skyline X in a d-dimensional numeric space $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_d)$, where $x_i \in [0, 1]$. The* ***bounding skyline point*** *of skyline X in dimension i, denoted by x_{Bound_i}, is defined as follows.*

$$x_{Bound_i}.\mathcal{D}_j = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases}$$

The bounding skyline points for two 2-dimensional skyline objects X and Y are marked in Figure 5.2 in red color. We remark that this concept is also applicable if the domain of one or more dimensions is not bounded. In this case the affected dimensions are bounded by the highest value that occurs in either skyline object X or Y in the according dimension. The coordinates of the remaining skyline points are then scaled respectively.

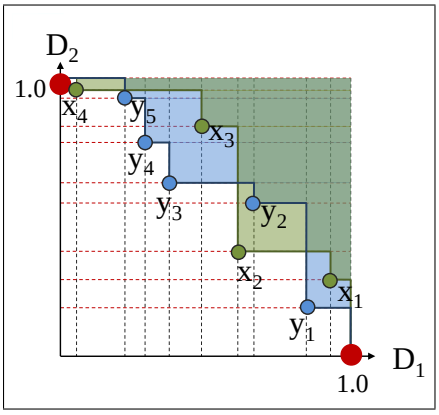


Figure 5.2: Dominance regions of two skylines X and Y.

5.4 Algorithms to Compute SkyDist

In this Section, we present two approaches for computing SkyDist. MC-SkyDist is based on a Monte-Carlo sampling approach whereas the second method exactly determines SkyDist by a plane-sweep based approach.

5.4.1 SkyDist by Monte-Carlo Sampling

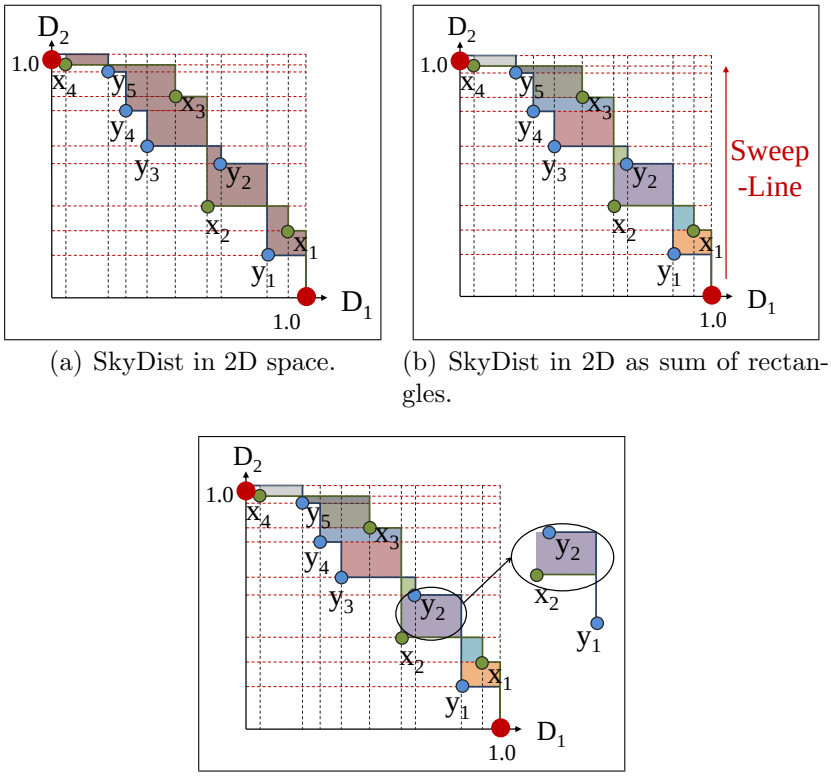
First we want to give an approximation of the distance of two skylines by a Monte-Carlo Sampling approach (MCSkyDist). Therefore, SkyDist is ap-

proximated by randomly sampling points and computing the ratio between samples that fall into the region defined by Equation 5.1 and the ones that do not. Let us consider Figure 5.3(a). The region marked in red illustrates the region underlying SkyDist of two Skylines X and Y. This region is the dominance region of skyline X and simultaneously the non-dominance region of skyline Y, thus the distance of skyline X to Y. A user defined number of points is randomly sampled and the amount of samples that fall into the SkyDist region is determined. The ratio between the samples located in the SkyDist region and the ones that do not give an approximation of the distance of the two skylines. We use this technique in our experiments as a baseline for comparing a more sophisticated approach.

5.4.2 SkyDist for 2-Dimensional Skylines

Now we describe the method of computing the exact distance of two skylines X and Y in 2-dimensional space. Let the skyline points of both skylines X and Y be ordered by one dimension, e.g. $\mathcal{D}_2$. Note that the dominance region of a skyline X is composed by the dominance regions of all of its skyline points. For the computation of SkyDist, we consider only the dominance regions that are exclusive for each skyline point. Meaning that when we look at a particular skyline point x_i, we assign the dominance region of x_i and discard all dominance regions of skyline points x_j, where $x_j.\mathcal{D}_2 > x_i.\mathcal{D}_2$.

Figure 5.3(a) illustrates in red the region underlying SkyDist according to skyline objects X and Y. This region can be considered as a sum of rectangles as illustrated in Figure 5.3(b). To calculate the region between the skylines X and Y and thus their distance we use the concept of a sweep-line approach. For this purpose we store the skyline points of both skylines X and Y in a heap structure called event point schedule (EPS), ordered by one of the two dimensions (e.g. $\mathcal{D}_2$) ascending. The general idea behind the sweep-line algorithm is that a line traverses across the plane, stopping at every point that is stored in the EPS. In our example, the sweep line moves



(a) SkyDist in 2D space.

(b) SkyDist in 2D as sum of rectangles.

(c) Exemplary computation of a rectangle area.

Figure 5.3: SkyDist Computation in 2-dimensional space.

along the axis of $\mathcal{D}_2$. Due to the ordering of the skyline points in the EPS we can determine the area of the rectangle at every stop of the sweep-line and calculate SkyDist in an incremental way. Figure 5.3(c) demonstrates for example the calculation process wenn the sweep-line holds on the skyline point y_2. Now we can calculate the area of the rectangle horizontal limited by the skyline points y_2 and x_2 as $(x_2.\mathcal{D}_1 - y_1.\mathcal{D}_1)(y_2.\mathcal{D}_2 - x_2.\mathcal{D}_2)$.

5.4.3 A Sweep-Plane Approach for the High-dimensional Case

In this Section, we describe how to exactly determine the SkyDist between skylines based on a sweep-plane paradigm referred to as SPSkyDist. We consider a d-dimensional skyline as a sequence of skylines with dimensionality $(d-1)$ (see Figure 5.4). Here, we use $\mathcal{D}_3$ (in decreasing order) as the ordering dimension and build a sequence of 2-dimensional skylines (each in the $\mathcal{D}_1/\mathcal{D}_2$-plane).

Traversal. The event point schedule (EPS) contains all points, ordered by the last coordinate (decreasingly). In each stop of the sweeping plane, we want to obtain a valid skyline in the space spanned by the remaining coordinates. This sub-skyline is stored in the sweep-plane status (SPS). In Figure 5.4, this refers to the four sub-skylines $X_1, ..., X_4$. More precisely, in each stop of the sweep-plane, this $(d-1)$-dimensional sub-skyline is updated by the following steps:

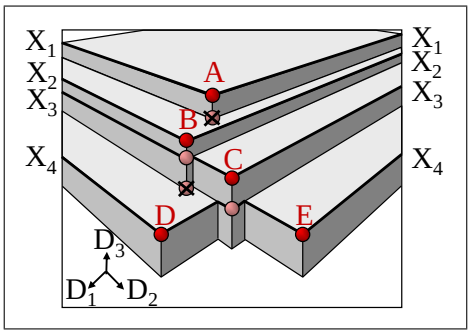


Figure 5.4: 3-dimensional skyline.

1. Projecting the current point of the EPS into the $(d-1)$-dimensional space,
2. Inserting the projected point into the skyline in the SPS,
3. Deleting those points from the skyline in the SPS which are dominated by the new point in the $(d-1)$-dimensional space,
4. Calling the traversal-algorithm recursively for the obtained $(d-1)$-dimensional skyline.

In our example, the EPS has the order (A, B, C, D, E). The method starts with an empty SPS, and at the first stopping point, the $\mathcal{D}_1/\mathcal{D}_2$-projection of A is inserted into the SPS to obtain X_1. No point is dominated at this stage. Hence, we call the traversal algorithm for the obtained 2-dimensional skyline A. At the next stop, the projection of B is inserted. Since the projection of B dominates the projection of A, X_2 only contains point B. After the recursive call, in the next stop, the projection of C is inserted which does not dominate object B in the skyline, and, therefore, the next skyline $X_3 = (B, C)$. Finally, D and E are inserted into the skyline in the SPS (which can be done in one single stop of the sweep-plane or in two separate stops in any arbitrary order), to obtain $X_4 = (D, C, E)$, since B is dominated by D in the $(d-1)$-dimensional projection.

Simultaneous Traversal for SkyDist. The computation of SkyDist(X, Y) requires a simultaneous traversal of the two skylines X and Y to be compared. Thus, the EPS contains the points of both X and Y, simultaneously ordered by the last coordinate. Each of the stops of the sweep-plane corresponds either to a point of X or a point of Y, and the corresponding sub-skyline in the SPS must be updated accordingly. Having developed this algorithmic template, we can easily obtain SkyDist(X, Y), because each stop of the sweep-plane defines a disklet, the thickness of which is given by the difference between the last and the current event point. This thickness must be multiplied with the $(d-1)$-dimensional volume which is returned by the recursive call that computes SkyDist of the $(d-1)$-dimensional sub-skylines X_i and Y_i. The volumes of all disklets of the obtained sub-skylines have to be added. This works no matter whether current or the previous event points belong to the same or different skylines or both skylines having identical values in the ordering coordinates or some points in the same skyline having identical values. In this case, some disklets with thickness 0 are added to the overall volume, and, therefore, the order in which the corresponding sub-skylines are updated, does not change anything.

5.5 Experiments

In this Section, we present an extensive experimental evaluation on synthetic and real world data. We demonstrate that SkyDist is highly effective and efficient. In order to demonstrate the potential of data mining on skylines for knowledge discovery, we integrated SkyDist into different clustering methods with different algorithmic paradigms. In particular, we selected the partitioning clustering approach PAM [94], the density-based algorithm DBSCAN [55] and Single Link [153] that forms a hierarchy of clusters. For skyline construction, the approach of [23] is applied.

5.5.1 Efficiency

To evaluate the stability of the baseline approach MCSkyDist and the scalability of both, MCSkyDist and SPSkyDist, we generate synthetic data of various number of objects and dimensions. Unless otherwise specified, the skyline is constructed from 1,000 uniformly distributed 2-dimensional data objects.

Accuracy of MCSkyDist w.r.t. Sample Rate. In the first experiment, we vary the sample rate in a range of 1 to 50,000 and quantify the accuracy of SkyDist in each run. Figure 5.5 indicates that the accuracy of MCSkyDist is very robust w.r.t. the number of samples. Its results achieve a constant value even with a small sample rate. Actually with 1,000 samples a constant SkyDist value can be achieved. Thus, we use a sample rate of 1,000 for MCSkyDist in the following experiments.

Runtime w.r.t. Number of Objects and Dimensionality. In order to measure the runtime for varying data set size and dimensionality of the data, we generate skylines of data sets with a size ranging from 10 to 10,000 points and dimensionality two, three and four. In most real world applications,

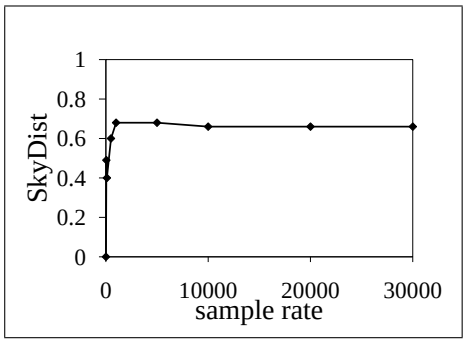


Figure 5.5: Varying sample rate of MCSkyDist.

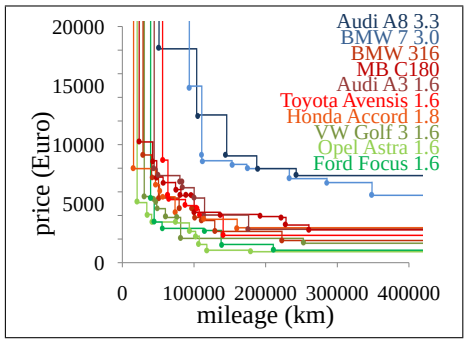


Figure 5.6: Clustering of car models represented by their skyline.

we are interested in the skyline w.r.t. a few selected dimensions. Hence, in this experiments, we focus on skylines up to dimensionality $d = 4$. All results are summarized in Table 5.1. In the 2-dimensional case the runtime of SkyDist remains constant even with increasing data size. This is due to the fact, that despite increasing database size the number of skyline points remains relatively constant. It has been shown in [108] that for independent distributed data the number of skyline objects is $O(\log_2 n)$.

Also in the 3- and 4-dimensional case it is evident that considering the skyline points instead of all data points is very efficient. It takes 78 and 266 ms for SPSkyDist and MCSkyDist respectively to return the result when comparing two skylines X and Y each determined out of 10,000 data points. MCSkyDist and SPSkyDist both scale linear with increasing dimensionality. However, the sweep-plane approach outperforms the baseline by a factor of two which confirms the effectiveness and scalability of an exact computation of SkyDist even for large data sets with more than two dimensions.

5.5.2 Clustering Skylines of Real World Data.

In addition to synthetic data we used real world data to demonstrate the potential of SkyDist for data mining. We demonstrate that interesting rea-

Table 5.1: Runtime analysis for SPSkyDist and MCSkyDist.

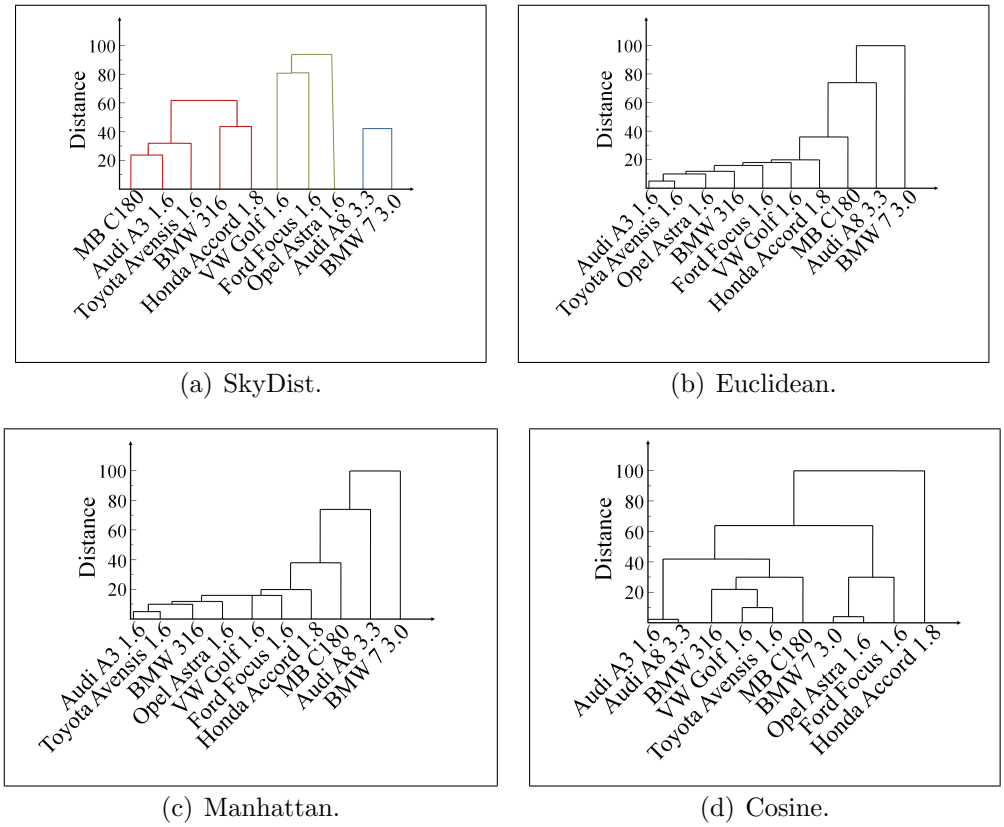
	# data points	# skyline points of X	# skyline points of Y	SPSkyDist	MCSkyDist
2D data	10	4	3	15 ms	47 ms
	100	5	6	16 ms	47 ms
	1000	9	5	15 ms	63 ms
	10000	7	12	15 ms	78 ms
3D data	10	5	5	31 ms	62 ms
	100	18	16	47 ms	109 ms
	1000	29	19	63 ms	125 ms
	10000	68	39	78 ms	266 ms
4D data	10	5	8	47 ms	78 ms
	100	25	36	172 ms	141 ms
	1000	80	61	203 ms	297 ms
	10000	187	151	609 ms	1078 ms

sonable knowledge can be obtained by clustering skylines with SkyDist. In particular, we focus on two case studies from different applications and we apply three different clustering algorithms (PAM, Single Link and DBSCAN) with SkyDist.

Case Study 1: Automotive Market. The data used in this experiment is obtained from the online automotive market place (`http://www.autoscout24.de`). The resulting data set comprises in total 1,519 used cars constructed in the year 2000. Thus, each data point represents a specific offer of a used car which are labeled to one of three classes *compact*, *medium-sized* and *luxury*, respectively. This information allows for an evaluation of the results. Each car model is represented by one 2-dimensional skyline using the

attributes *mileage* and *price*. Hence, each skyline point represents a specific offer that provides an outstanding combination of these attributes, and therefore it is interesting for the user. PAM, DBSCAN and Single Link combined with SkyDist create an identical clustering result (cf. Figure 5.6) using the following parameterization: PAM ($k = 3$), DBSCAN ($\epsilon = 10$, $MinPts = 2$) and the dendrogram Single Link with a vertical cut at $maxDistance = 90$. All algorithms produce 100% class-pure clusters w.r.t. the labelling provided by the online market place. As expected the Audi A8 and BMW 7 of class *luxury* are clustered together in the blue cluster with a very low distance. Also the Mercedes Benz C180 (MB C180) and the Audi A3 belong to a common cluster (marked in red) and show a larger distance to the Toyota Avensis or the Honda Accord. These two car models are clustered together in the green cluster, whose members usually have a cheaper price. The clustering result with SkyDist is very informative for a user interested in outstanding combinations of *mileage* and *price* but not fixed on a specific car model. By clustering, groups of models with similar skylines become evident.

SkyDist vs. Conventional Metrics. Conventional metrics can in principle be applied for clustering skylines. For this purpose, we represent each car model as a vector by calculating the average of the skyline points of the respective car model in each dimension. Then we cluster the resulting vectors with Single Link using the Euclidean, Manhattan and Cosine distance. In contrast to clustering skylines using SkyDist (cf. Figure 5.7(a)), Figures 5.7(b), 5.7(c) and 5.7(d) demonstrates that no clear clusters are identifiable in the dendrogram and the result is not very comprehensible. In Figure 5.7(b) and 5.7(c) it can easily be seen that Euclidean and Manhattan distance lead to similar results. Both show the so called Single Link effect, where no clear clusters can be identified. Using the Cosine distance avoids this effect but does not produce meaningful clusters either. For example, the luxury car model BMW 7 has minimum distance to the Opel Astra



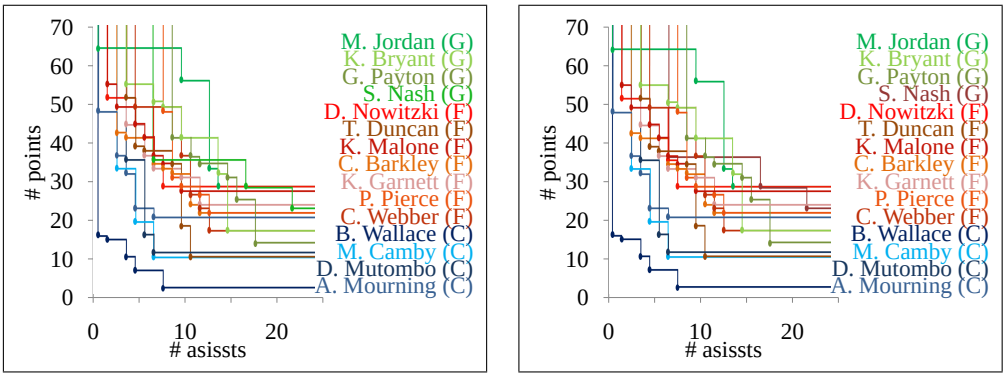
(a) SkyDist. (b) Euclidean.

(c) Manhattan. (d) Cosine.

Figure 5.7: The dendrogram of Single Link using SkyDist in comparison to conventional metrics.

of class *compact* but has an unexpected high distance to the luxury Audi A8.

Case Study 2: Performance Statistics of Basketball Players. The NBA game-by-game technical statistics are available at `http://www.NBA.com`. We focus on the years 1991 to 2005. To facilitate demonstration and interpretation, we select players who have played at minimum 500 games and are noted for their skills and got various awards. The players are labeled with the three different basketball positions *guard* (G), *forward* (F) and *center* (C). The individual performance skyline of each player represents the number of assists and points. We cluster the skylines using SkyDist.



(a) Single Link, DBSCAN. (b) PAM

Figure 5.8: Clustering of NBA Players represented as skylines.

Single Link with a vertical cut of the dendrogram at $maxDistance = 96$ and DBSCAN ($\epsilon = 4$, $MinPts = 2$) result in the same clustering. Figure 5.8(a) shows that the players cluster very well in three clusters that refer to the labels G, F and C. With PAM ($k = 3$) (cf. Figure 5.8(b)) only Steve Nash (G) clusters into the red *forward* cluster. This can be explained by the fact, that this player has performance statistics as a player in the position forward concerning number of points and assists.

Part III

Techniques for Mining Biomedical Data

Chapter 6

Automated Detection of Brain Atrophy Patterns based on MRI

Magnetic resonance imaging (MRI) allows to display brain structures with highest resolution. To fully exploit the potential of this imagining modality, data mining methods are required to reveal subtle differences in brain structure caused by disorders such as Mild Cognitive Impairment (MCI) and early stage Alzheimer's disease (AD). For a concise report on MRI please refer to Chapter 2.5. In this Chapter, we present a data mining framework which combines elements from feature selection, clustering and classification in order to discriminate between diseased and healthy subjects.

The Chapter is organized as follows. After an introduction in the next Section, we report related work in this field. In Section 6.2 we present the framework combining feature selection, clustering and classification. The classification accuracies and visualization of the affected areas are presented in Section 6.3. In Section 6.4 we discuss the results. The concepts described in this Chapter have been published in [141].

6.1 Introduction

Alzheimer's disease (AD) is the most frequent cause of age-related dementia. Due to the increasing proportion of elderly people in the Western societies, the prevalence of dementia is projected to double within the next three decades [62]. The reliable and early detection of AD in pre-dementia stages such as mild cognitive impairment (MCI) is the basis for the development of preventive treatment approaches. However, especially the diagnosis of mild AD and prediction of development of AD in at-risk groups remains challenging. In addition to cerebrospinal fluid derived markers [17, 56, 77, 80, 180], neuroimaging markers have been recommended to be included in the revised NINCDS-ADRDA diagnostic standard criteria [50] and proposed as predictors of AD [137, 174]. The best established MRI derived marker of AD, hippocampus volume, shows relatively high diagnostic accuracy for AD but clinically insufficient predictive value for the prediction of progression from MCI to AD when assessed as the sole predictor [39, 87, 90, 95, 134, 157, 169]. As an alternative to ROI (Region of interest) based volumetry, automated morphometry and deformation-based approaches have been developed to map the pattern of structural brain changes across the entire brain [8, 72]. A series of voxel-based morphometric studies in MCI and mild AD have shown marked volume differences not only within the hippocampus area but also distributed within cortical brain areas such as the precuneus and cingulate gyrus [12, 31, 32, 64, 92, 135]. However, few statistical approaches have been proposed to derive individual risk scores from such maps of atrophy for the clinical prediction of AD. Data mining approaches and pattern recognition methods provide a way to extract from millions of voxels within an MRI the minimal set of voxel values necessary to attain a sufficiently high accuracy for the prediction and diagnosis of AD. Multivariate approaches such as principal component analysis (PCA) [65], independent component analysis (ICA) [114], structural equation modeling [113], and support vector machine [123, 124] are potential candidates but have mostly been applied to

functional neuroimaging data so far. Recently, such multivariate methods have been adopted for the analysis of structural MRI to detect spatial patterns of atrophy in AD [30, 40, 43, 51, 52, 53, 57, 96, 119, 162, 168]. These techniques allow for deriving a single value representing the degree to which a disease specific spatial pattern of atrophy is present in a single individual. The application of such classifiers of spatial pattern of atrophy in MCI has shown promising results for the prediction of AD [40, 162]. We applied a novel two-step approach combining a distribution-free feature selection algorithm at the first stage and, at the second stage, different multivariate classifiers for case-by-case decision making. The major aims were, first, to develop a novel feature selection method to circumvent potential problems of previous approaches for feature selection including lack of statistical power due to multiple testing [57] or purely-data driven correlational patterns in unsupervised dimensionality reduction (e.g. PCA [162, 163]). Secondly, we compared different cross-validated classifiers including support vector machine (SVM), a Bayesian classifier, and voting feature intervals (VFI) combined with unsupervised clustering algorithms to derive the minimal set of voxels for optimized prediction of diagnosis (AD vs. HC) or prediction of AD in MCI. The overall goal was to derive an optimized classification that is sensitive for the early MRI-based detection of AD.

6.2 The FCC framework

We applied a multi-step data mining procedure including feature selection, clustering and classification (FCC) to identify the best discriminating regions in brain images. Given a data set DS consisting of MRI scans of n subjects $s_1, \cdots, s_n$ labeled to a set of k discrete classes $C = \{c_1, \cdots, c_k\}$ (in our study e.g. HC and AD), we denote the class label of subject s_i by $s_i.c$. For each subject we have an MR image which is represented as a feature vector V composed of d voxels $v_1, \cdots, v_d$.

6.2.1 Feature Selection

First, we select the most discriminating features using a feature selection criterion. We use the Information Gain [146] to rate the interestingness of a voxel for class separation, which requires the following definitions.

Entropy of the Class Distribution. The entropy of the class distribution $H(C)$ is defined as $H(C) = \sum_{c_i \in C} p(c_i) \cdot log_2(p(c_i))$, where $p(c_i)$ denotes the probability of class c_i, i.e $|\{s|s \in DS \wedge s.c = c_i\}|/n$. $H(C)$ corresponds to the required amount of bits to tell the class of an unknown subject and scales between 0 and 1. In the case of $k = 2$, (e.g. we consider the two classes HC and AD), if the number of subjects per class is equal for both classes, $H(C) = 1$. In the case of unbalanced class sizes the entropy of the class distribution is smaller than one and approaches zero if there are much more instances of one class than of the other class.

Information Gain of a Voxel. Now we can define the Information Gain (IG) of a voxel v_i as the amount by which $H(C)$ decreases through the additional information provided by v_i on the class, which is described by the conditional entropy $H(C|v_i)$.

$$IG(v_i) = H(C) - H(C|v_i)$$

In the case of k=2, the Information Gain scales between 0 and 1, where 0 means that the corresponding voxel provides no information on class label of the subjects. An Information Gain of 1 means that the class labels of all subjects can be derived from the corresponding voxel without any errors. To compute the conditional entropy, features with continuous values, as in our case, need to be discretized using the algorithm of [59]. This method aims at dividing the attribute range into class pure intervals. The cut points are determined by the Information Gain of the split. Since a higher number of

cut points always implies higher class purity but may lead to over fitting, an information-theoretic criterion based on the Minimum Description Length principle is used to determine the optimal number and location of the cut points.

6.2.2 Clustering

After feature selection, we apply a clustering algorithm to identify groups of adjacent voxels with a high discriminatory power and to remove noise. Clustering algorithms aim at deriving a partitioning of a data set into groups (clusters) such that similar objects are grouped together. We apply clustering to group voxels with similar spatial location in the brain and similar (high) IG. The density-based clustering algorithm DBSCAN [55] has been designed to find clusters of arbitrary shape in databases with noise. In our context, clusters are connected areas of voxels having a high IG which are separated by areas of voxels of lower IG. DBSCAN has been originally designed for clustering data objects represented by feature vectors. A detailed description of DBSCAN is provided in Section 2.1. To adapt the algorithm to our setting, we redefine the core object property and direct density reachability as follows:

Given two thresholds of Information Gain t_{core} and t_{border} and a minimum number of voxels $MinVox$ we call a voxel v_i a core voxel if the IG of v_i is larger than t_{core} and v_i is surrounded by at least $MinVox$ voxels having an IG of at least t_{border}. We allow for potentially different thresholds $t_{core} > t_{border}$ of IG for core voxels and voxels at the boundaries of the clusters to require highly discriminative cluster centers and to model the natural fading of the discriminatory power in the boundary areas of the clusters. However, it is on our specific set of images not necessary to distinguish between t_{core} and t_{border} since the voxels either have a significant Information Gain value or an IG of zero. So we set t_{core} and t_{border} to the minimum IG in the data set and used $MinVox = 6$, which means that we require a core voxel to be situated in a neighborhood of highly discriminative voxels.

6.2.3 Classification

After clustering, the selected features represent spatially coherent regions which exhibit significant differences among the groups. At this stage, classification algorithms can be applied to validate the discriminatory power of these selected clusters. Classification is a data mining technique used to predict group membership for data instances, which are the subjects in our application. The task of classification involves two major steps: In the so-called training phase, the classifier learns the separating information. To achieve this, some amount of instances with known class labels is required. In the test phase, the classifier predicts the class label of unlabeled instances based on the learned information. For more information on the validation of classifiers see Section 2.4. Among the large variety of classifiers we chose three representative approaches with very different algorithmic paradigms.

- **Linear Support Vector Machine (SVM).** SVM aims at constructing a hyperplane separating the training examples. Among all possible hyperplanes, SVM selects the one with the maximum margin between the training examples of both classes.

- **Bayesian Classifier (Bayes).** The fundamental idea of Bayesian classification is to model each class of the training data by a probability density function. Test objects are then assigned to most probable class.

- **Voting Feature Intervals (VFI).** Very different to SVM and Bayes, VFI is a simple entropy-based classifier. In the training phase, VFI constructs class-pure intervals for each feature and each class. Classification is performed by voting.

For more information on the classifiers, please refer to Section 2.3.

6.2.4 Visualization

We display the spatial location of the features best discriminating the classes HC and AD, and MCI-MCI and MCI-AD, respectively. For the ease of comparison, we display in Figures 6.1, 6.2(b) and 6.3(b) the features which are relevant for classification in all folds. Note that this is only done to obtain one common spatial map for interpretation of the best discriminating regions and the reported classification accuracies are obtained by leave-one-out cross-validation. To facilitate interpretation, we additionally highlight the most interesting clusters in different colors in Figures 6.2 and 6.3. We were interested in clusters which are as large as possible and exhibit an IG as high as possible. Therefore, we selected those clusters in the visualization exhibiting an outstanding combination of both criteria using the skyline operator which has been successfully applied in many multi-criteria decision making applications, e.g. in personalized information systems [82] or for the selection of web services [154]. The skyline of a data set consists of all data objects which are not dominated by any other object in the data set w.r.t. any possible weighting of the studied criteria. We have already given a detailed overview on skyline computation in Chapter 5. In our context, we consider clusters of voxels which are described by the features size and IG. To get the anatomical information of the clusters, we used the Talairach Daemon software [1] after MNI to Talairach coordinate transformation with the non-linear approach by Duncan *et al.* [54].

6.3 Experiments

We applied the FCC framework to a set of MRI scans of patients with amnestic MCI, AD and healthy controls.

[1] `http://www.talairach.org/`

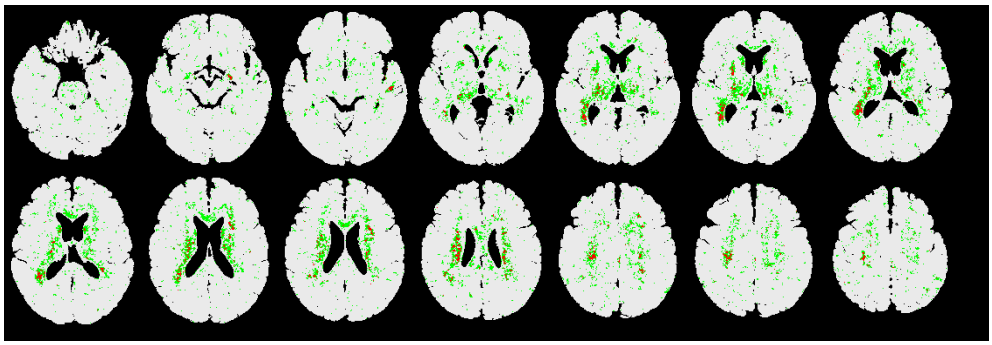


Figure 6.1: Selected features for the comparison between AD vs. HC. z-coordinates in Talairach space: top row of images -45.5, -33.5, -26.5, -18.5, -13.5, -11.5, -5.5; bottom row: -3.5, 0.5, 4.5, 8.5, 13.5, 15.5, 21.5.

6.3.1 Subjects

32 patients with clinically probable AD, 24 patients with amnestic MCI and 18 healthy control subjects (HC) underwent MRI and clinical examinations (cf. Table 6.1). AD patients fulfilled the criteria of the National Institute of Neurological Communicative Disorders and Stroke and the Alzheimer Disease and Related Disorders Association (NINCDS-ADRDA) criteria for clinically probable AD [115]. MCI subjects fulfilled the Mayo criteria for amnestic MCI [137]. All MCI subjects had subjective memory complaints, a delayed verbal recall score at least 1.5 standard deviations below the respective age norm, normal general cognitive function, and normal activities of daily living. Severity of cognitive impairment was assessed by the Mini-Mental-State-Examination (MMSE) [63]. Controls did not have cognitive complaints and scored within 1 standard deviation from the age adjusted norm on all subtests of the CERAD cognitive battery [122]. MCI patients received clinical follow-up examinations over approximately 2.5 years, using clinical examination and neuropsychological testing to determine which subjects converted to AD and which remained stable.

Table 6.1: Demographic variables and MMSE for the different groups.

Group	Women/men	Age in years)	MMSE mean
Healthy controls	9/9	64.8	29.3
AD patients	20/12	68.8	23.4
MCI patients	13/11	69.7	27.0

6.3.2 MRI Acquisition

MRI examinations of the brain were performed on a 1.5 T MRI scanner (Magnetom Vision, Siemens Medical Solutions, Erlangen, Germany). We acquired a high-resolution T1-weighted Magnetisation Prepared Rapidly Acquired Gradient echo (MPRAGE) 3D-sequence with a resolution of 0.55 by 0.55 by 1.1 mm^3, TE = 3.9 ms, TI = 800 ms, and TR = 1,570 ms. The FOV was 240 mm and the pixel matrix was 512 x 512.

6.3.3 MRI Processing

The preprocessing of the scans was conducted with the statistical software package SPM2 [1]. The high-dimensional normalization of the MRI scans was processed according to a protocol that has been described in detail previously in [162] and [163]. First, we constructed a customized template across groups averaged across images that were normalized to the standard MNI T1 MRI template, using the low-dimensional transformation algorithm implemented in SPM2[[8], [7]]. Next, one good quality MRI scan of a healthy control subject was normalized to this anatomical average image using high-dimensional normalization with symmetric priors [6] resulting in a pre-template image.

[1]Wellcome Trust Centre for Neuroimaging, London, `http://www.fil.ion.ucl.ac.uk/spm/`

Finally, the MRI scan in native space of the same subject was normalized to this pre-template image using high-dimensional normalization. The resulting volume in standard space served as the anatomical template for subsequent normalizations of the remaining scans. The individual anatomical scans in standard space (after low-dimensional normalization) were normalized to the anatomical template using high-dimensional image warping [6]. These normalized images were resliced to a final isotropic voxel size of 1.0 mm^3. Finally, we derived Jacobian determinant maps from the voxel-based transformation tensors. Values above 1 represent an expansion of the voxel, values below 1 a contraction of the voxel from the template to the reference brain. The resulting Jacobian determinant maps were masked for brain matter and cerebrospinal fluid (CSF) spaces using masks from the segmented template MRI [7]. To obtain the brain mask, the template brain scan was segmented into gray and white matter and CSF spaces. The gray matter (GM) and white matter (WM) compartments then were combined to obtain a brain mask excluding CSF: $(GM+WM)/(WM+GM+CSF)$.*BRAIN with gray matter, white matter, and CSF representing the gray and white matter and CSF probabilistic maps obtained through segmentation and BRAIN representing the brain mask obtained from the brain extraction step in SPM2. We took the logarithm of the masked maps of the Jacobian determinants [151] and then applied a 10 mm full width at half maximum isotropic Gaussian kernel. The masked smoothed Jacobian determinant maps were scaled to the same mean value and standard deviation using a voxel-wise z-transformation:

$$z_{i,k} = \frac{x_{i,k} - \overline{x}_k}{s_k}$$

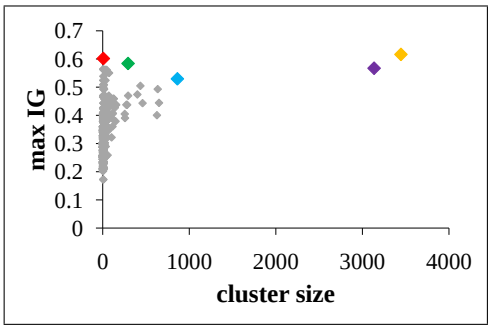
where $x_{i,k}$ is the FA value of voxel i in scan k, $\overline{x}_k$ is the mean value across all x_i of scan k and s is the standard-deviation across all x_i of scan k.

6.3.4 Results

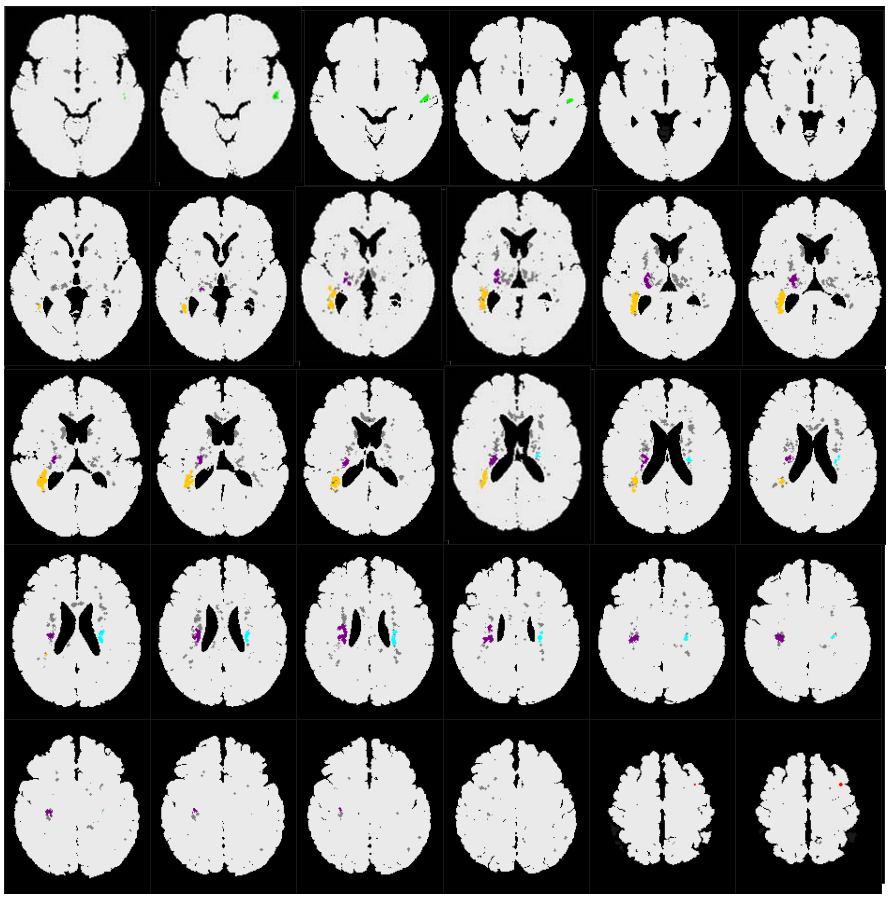
The mean age, MMSE and the gender distribution for AD, MCI, and HC subjects are displayed in Table 6.1. Nine out of 24 MCI subjects converted to AD after an average follow-up interval of 2.5 years. We applied our framework with leave-one-out cross validation on three data sets to identify highly selective brain regions for the differentiation between AD vs. HC, MCI vs. HC, and MCI converter (MCI-AD) vs. MCI non-converters (MCI-MCI). For the train-and-test validation, we used the brain regions identified in the AD vs. HC for the prediction of conversion of patients with MCI. Table 6.2 summarizes all experiments performed and Table 6.3 provides a summary of the classification results for all group comparisons, which are explained below in detail. For all classification results, also the 95% confidence interval is provided in Table 6.3.

Table 6.2: Summary of classification experiments.

Task	Comparison	Validation	Training	Test
1	AD vs. HC	Leave-one-out	n.a.	n.a.
2	MCI-MCI vs. MCI-AD	Leave-one-out	n.a.	n.a.
3	MCI vs. HC	Leave-one-out	n.a.	n.a.
4	MCI-MCI vs. MCI-AD	Train-and-Test	AD vs. HC	MCI



(a) Cluster size and maximum Information Gain for MCI converter vs. MCI non-converter.



(b) Selected features after HC vs. AD clustering. Colors: Cluster 1 red, cluster 2 green, cluster 3: blue, cluster 4 purple cluster 5 orange. remaining clusters gray. Displayed is every second slice starting with z= -31.5 to 22.5; 34.5 and 35.5.

Figure 6.2: AD vs. HC. Discriminating regions in the brain with high IG value.

Task	Measure	SVM		Bayes		VFI	
1	Accuracy	90%	[77.41, 96.26]	92%	[79.89, 97.41]	78%	[63.67, 88.01]
	Sensitivity	96.88%	[82.01, 99.84]	93.75%	[77.78, 98.27]	65.63%	[46.78, 80.83]
	Specificity	77.78%	[51.92, 92.63]	88.89%	[63.93, 98.05]	100%	[78.12, 100]
2	Accuracy	95.83%	[76.88, 99.78]	91.67%	[71.53, 98.54]	95.83%	[76.88, 99.78]
	Sensitivity	88.89%	[50.67, 99.42]	77.78%	[40.19, 96.05]	100%	[62.88, 100]
	Specificity	100%	[74.65, 100]	100%	[74.65, 100]	93.33%	[66.03, 99.65]
3	Accuracy	97.62%	[85.91, 99.88]	85.71%	[70.76, 94.05]	88.1%	[73.57, 95.54]
	Sensitivity	95.83%	[76.88, 99.78]	83.33%	[61.81, 94.52]	83.33%	[61.81, 94.52]
	Specificity	100%	[78.12, 100]	88.89%	[63.93, 98.05]	94.44%	[70.62, 99.71]
4	Accuracy	50%	[29.65, 70.35]	58.33%	[28.99, 81.38]	75%	[52.95, 89.40]
	Sensitivity	55.56%	[22.26, 84.66]	46.66%	[22.22, 72.57]	55.56%	[22.66, 84.66]
	Specificity	46.47%	[22.28, 72.58]	77.77%	[40.19, 96.05]	86.67%	[58.39, 97.66]

Table 6.3: Classification Results. For all classifiers and experiments, accuracy, sensitivity and specificity are provided together with the 95% confidence intervals

Classification of AD vs. HC. For the differentiation between AD and HC, a proportion of the voxels ranging between 97.48% and 98.04% have an Information Gain of 0, i.e. they contain no information separating the groups and are therefore excluded from further analysis. Theoretically, combinations of these features may provide valuable information. However, due to the high dimensionality of the data, an exhaustive search for feature combinations is not applicable.

For one randomly selected fold the range of IG value among the remaining 87,416 voxels was between 0.18 and 0.69. The minimum IG of 0.18 was relatively high, indicating that the voxels either contain a good deal of valuable information to separate the classes or are completely irrelevant. Figure 6.1 displays the spatial distribution of the voxels with non-zero IG across all folds.

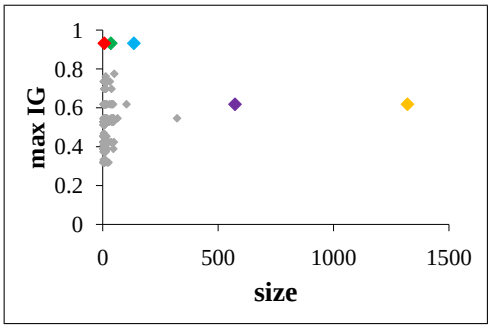
For one randomly selected fold clustering reduced the 87,416 selected features to 26,228. In total, 978 clusters containing at least one core object exhibiting the maximum number of six neighbors were obtained. The largest cluster comprised 3,445 voxels. Figure 6.2(a) summarizes the cluster

Table 6.4: Clusters AD vs. HC.

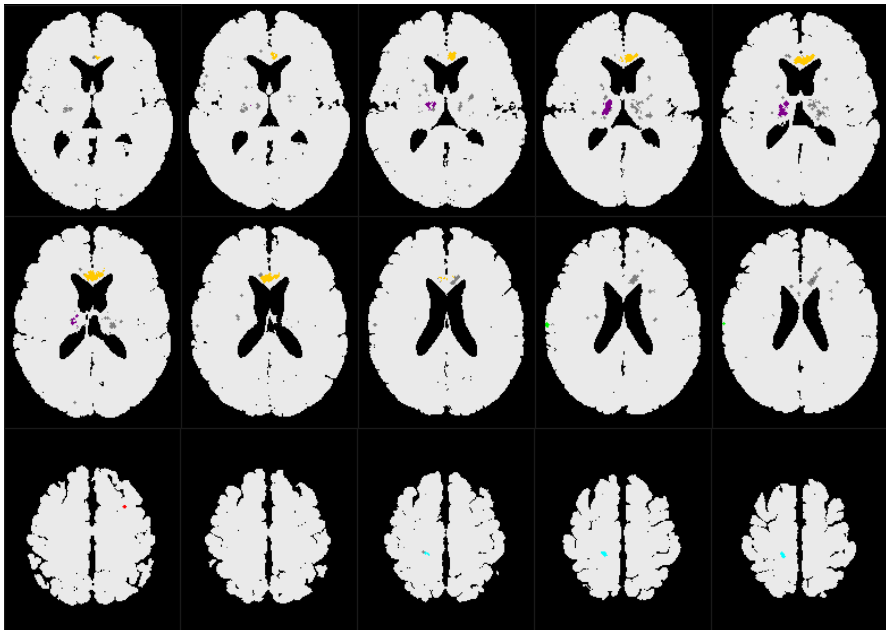
Cluster-ID	Size(#voxels)	Max IG	Location	Regions
5 (orange)	3,445	0.62	41.58, 28.28, -16.98	Frontal Lobe, Inferior Frontal Gyrus, White Matter
			40.59, 33.42, -11.34	Frontal Lobe, Middle Frontal Gyrus, Gray Matter, Brodmann area 11
			34.65, 16.96, -10.52	Frontal Lobe, Extra-Nuclear, Gray Matter, Brodmann area 47
			42.57, 31.32, -14.61	Frontal Lobe, Inferior Frontal Gyrus, Gray Matter, Brodmann area 11
			34.65, 24.47, 3.84	Frontal Lobe, Inferior Frontal Gyrus, Gray Matter, Brodmann area 45
			41.58, 26.31, -17.72	Frontal Lobe, Inferior Frontal Gyrus, Gray Matter, Brodmann area 47
			41.58, 11.02, -12.75	Sub-lobar, Extra-Nuclear, Gray Matter, Brodmann area 13
			37.62, 22.05, -5.73	Sub-lobar, Extra-Nuclear, Gray Matter, Brodmann area 47
			34.65, 12.41, -4.41	Sub-lobar, Insula, Gray Matter, Brodmann area 13
			34.65, 17.38, -2.13	Sub-lobar, Insula, Gray Matter, Brodmann area 47
			41.58, 11.94, -13.64	Temporal Lobe, Superior Temporal Gyrus, Gray Matter, Brodmann area 38
4 (purple)	3,135	0.57	23.76, -4.36, -9.45	Limbic Lobe, Parahippocampal Gyrus, Gray Matter, Amygdala
			24.75, -0.61, -12.16	Limbic Lobe, Parahippocampal Gyrus, Gray Matter, Brodmann area 34
			26.73, 2.34, -11.47	Limbic Lobe, Subcallosal Gyrus, Gray Matter, Brodmann area 34
			33.66, -4.45, 8.05	Sub-lobar, Claustrum, Gray Matter
			29.7, -6.30, 9.99	Sub-lobar, Lentiform Nucleus, Gray Matter, Putamen
			32.67, 7.31, 10.23	Sub-lobar, Claustrum, Gray Matter
			32.67, 8.37, 12.02	Right Cerebrum, Sub-lobar, Insula, Gray Matter, Brodmann area 13
			24.75, -6.25, -8.52	Sub-lobar, Lentiform Nucleus, Gray Matter, Lateral Globus Pallidus
			25.74, -4.32, -8.62	Sub-lobar, Lentiform Nucleus, Gray Matter, Putamen
			22.77, 9.92, -15.21	Frontal Lobe, Inferior Frontal Gyrus, Gray Matter, Brodmann area 47
			25.74, 4.19, -13.24	Frontal Lobe, Subcallosal Gyrus, Gray Matter, Brodmann area 34
3 (blue)	862	0.52	-24.75, -0.76, 4.18	Sub-lobar, Lentiform Nucleus, Gray Matter, Putamen
			-23.76, 6.34, 10.28	Sub-lobar, Extra-Nuclear, White Matter
			-26.73, 11.09, 8.20	Sub-lobar, Claustrum, Gray Matter
			-19.8, 0.9058, -1.30	Sub-lobar, Lentiform Nucleus, Gray Matter, Lateral Globus Pallidus
2 (green)	293	0.58	-49.5, -3.13, -23.81	Temporal Lobe, Fusiform Gyrus, Gray Matter, Brodmann area 20
			-50.49, -1.15, -23.07	Temporal Lobe, Middle Temporal Gyrus, Gray Matter, Brodmann area 21
1 (red)	7	0.59	-33.66, -23.51, 34.81	Parietal Lobe, Postcentral Gyrus, Gray Matter, Brodmann area 2

Table 6.5: Clusters MCI-AD vs. MCI-MCI.

Cluster-ID	Size(#voxels)	Max IG	Location	Regions
5 (orange)	1,320	0.61	-1.98, 47.87, -5.59	Anterior Lobe, Culmen, Gray Matter
4 (violet)	573	0.62	15.84, -0.27, -5.45	Sub-lobar, Lentiform Nucleus, Gray Matter, Medial Globus Pallidus
			15.84, 1.66, -5.55	Sub-lobar, Lentiform Nucleus, Gray Matter, Lateral Globus Pallidus
			14.85, -7.85, -1.71	Sub-lobar, Extra-Nuclear, White Matter
			19.80, 3.69, -3.97	Sub-lobar, Lentiform Nucleus, Gray Matter, Putamen
3 (blue)	135	0.93	16.83, 14.50, 37.50	Frontal Lobe, Cingulate Gyrus, Gray Matter, Brodmann area 32
			18.81, 15.33, 34.67	Frontal Lobe, Cingulate Gyrus, White Matter
			20.79, 16.34, 35.57	Frontal Lobe, Sub-Gyral, White Matter
			18.81, 16.26, 33.73	Limbic Lobe, Cingulate Gyrus, White Matter
			14.85, 19.39, 38.18	Limbic Lobe, Sub-Gyral, White Matter
2 (green)	35	0.93	67.32, -0.67, 6.02	Temporal Lobe, Superior Temporal Gyrus
1 (red)	7	0.93	-27.72, -23.65, 32.04	Frontal Lobe, Sub-Gyral, White Matter



(a) Cluster size and maximum Information Gain for MCI converter vs. MCI non-converter.



(b) Skyline clusters of MCI-AD vs. MCI-MCI. Colors: cluster 1: red, cluster 2: green, cluster 3: blue, cluster 4: purple, cluster 5 orange. Displayed are some representative slices containing clusters: z-coordinates in Talairch space: -12.5 to 5.5 and 34.5 to 42.5 (every second slice).

Figure 6.3: MCI-AD vs. MCI-MCI. Discriminating regions in the brain with high IG value.

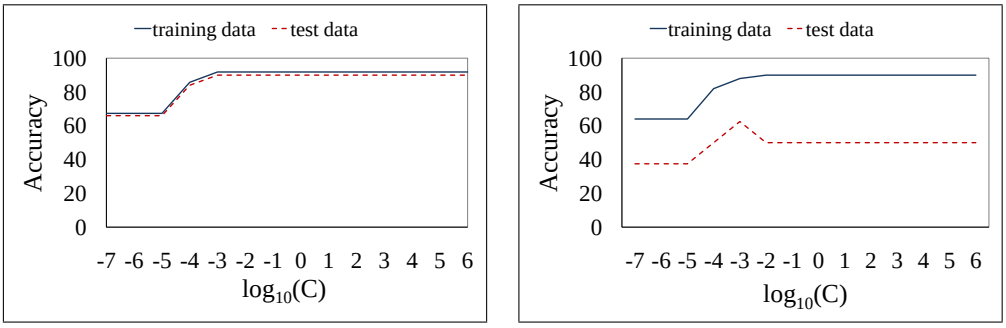
statistics with respect to the two most important criteria: the size of the clusters and value of IG. The anatomical locations of the skyline clusters have been highlighted with the same colors in Figure 6.2(b). Table 6.4 provides a summary of the clusters including the anatomical location. Due to space limitation, only the anatomical location of the best separating regions with an IG of at least 0.3 for the large clusters 3 to 5 are included in Table 6.4. The clusters were centered within the medial temporal lobe including the hippocampus, parahippocampus, amygdala, adjacent basal ganglia, the right anterior cingulate gyrus extending towards the prefrontal cortex, left insula, and claustrum (Table 6.4). On the basis of the selected clusters, a classification accuracy of 92% with Bayes (sensitivity: 94%, specificity: 89%), 90% with SVM (sensitivity: 97%, specificity: 78%) and 78% with VFI (sensitivity: 66%, specificity: 100%) was obtained (cf. classification task 1 in Table 6.2).

Classification of MCI-AD vs. MCI-MCI. When applying feature selection on the brain images of the group of MCI with respect to conversion between 97.82% and 98.73% of the voxels have an Information Gain of 0. For one randomly selected fold we obtained 74,680 features with IG greater than zero. The minimum occurring IG was 0.32. Clustering reduced the number of features to 10,775. The selected clusters separated converters and non-converters with high accuracy: 95.83% accuracy was obtained with SVM and VFI (sensitivity SVM: 89%, VFI 100%, specificity SVM 100%, VFI 93%), cf. task 2 in Table 6.2. With Bayes, an accuracy of 91.67% has been obtained (sensitivity: 78% specificity: 100%). The skyline clusters for the separation of converters and non-converters are displayed in Figure 6.3(a) and 6.3(b). The corresponding anatomical regions are provided in Table 6.5. In total, 276 clusters were obtained.

Using AD vs. HC as training data and MCI as test data, best results have been obtained with VFI. Conversion was predicted with an accuracy of

75% (sensitivity: 56%, specificity: 87%).

When contrasting the group of subjects with MCI against the HC for one randomly selected fold, a total of 37,504 characteristic features were obtained. Clustering reduced the number of features to 2,190. On the clustered data linear SVM performed best with 97.62% accuracy (sensitivity: 96%, specificity: 100%), cf. task 3 in Table 6.2. With VFI, we obtained an accuracy of 88.1% (sensitivity: 83%, specificity: 94%), and with Bayes an accuracy of 85.71% (sensitivity: 83%, specificity: 88%). The spatial location of the best discriminating clusters was similar to that of MCI-MCI vs. MCI-AD and therefore was not displayed.



(a) Task 1 (b) Task 4

Figure 6.4: Effect of the parameter C on the classification accuracy of SVM in task 1 (a) and task 4 (b). For both tasks we can observe only minor influence of C for very small C ($log_{10}(C) < -3$).

Parameter Settings for Classification. An important parameter for SVM is the complexity constant C. For the soft margin SVM, the parameter $C > 0$ determines the trade-off between margin maximization and training error minimization. To systematically investigate the influence of the complexity constant on the classification accuracy, we repeated the experiments with various settings for C in the range of $log_{10}(C) = -7$ to $+6$. The analysis was applied to 2 out of the 4 classification tasks: task 1 which

involves the classification of AD vs. HC with leave-one-out cross-validation and task 4 which involves the prediction of conversion in MCI based on the model learned from AD vs. HC with train-and-test validation. We selected those two tasks for two reasons: SVM is outperformed by other classifiers in these settings (see Table 6.3) and the classification accuracy of SVM varies, depending upon the validation procedure applied in task 1 and 4.

The classification result of task 1 for various settings of C is displayed in Figure 6.4(a). Since task 1 included a leave-one-out cross-validation, the accuracy on both training and test data, respectively, were averaged among all folds. As can be seen in figure 6.4, the parameter C had only very minor influence on the classification accuracy. The classification accuracy for the training data (90%) and the accuracy for the test data (91.84%) were constant for a wide range of parameter settings (of $log_{10}(C) = -3$ to 6). Only for a very small C, the classification accuracies for both the training and test data sets decreased numerically ($log_{10}(C) < -3$). For $log_{10}(C) < -5$ we observed a trend towards a statistically significant decrease in classification accuracy compared to the optimal level, i.e. the accuracy was 66% (95%CI = [52.15, 77.56]) for $log_{10}(C) < -5$ vs. 90% (95%CI [77.41, 96.26]) for $log_{10}(C) = -3$ to 6 when applied to the test data set. Figure 6.4(b) displays the result of an analogous analysis for task 4. Note that this is the most difficult classification task since the training and the test data stem from different groups of subjects. Thus, the whole data mining pipeline including feature selection, clustering and training of the SVM is applied to the training data AD vs. HC. Based on the learned model, the SVM predicts the conversion of test subjects with MCI. Figure 6.4(b) displays the accuracy for the training data, i.e. the accuracy of the SVM applied to AD vs. HC for varying settings of the complexity constant C. In addition, the accuracy for the test data is displayed, i.e. the accuracy to predict conversion in MCI subjects. We can observe two different aspects from Figure 6.4(b). First, the training data could be well separated using a linear kernel. Since more complex kernels

(like polynomial, radial basis, etc.) lead more likely to overfitting and thus lower classification accuracy when applied to the test data, their application is indicated only in the case when the training data cannot be well separated using the linear kernel. Second, visual inspection of Figure 6.4(b) shows that the choice of the complexity parameter C had only very minor influence on the accuracy. Within the range of $log_{10}(C) = -3$ to 6, the classification accuracy for the training data was constantly 90% and the accuracy for the test data was constantly 50%. Similar to the results for task 1 (see above), the classification accuracy for both the training and test data decreased only for very small C values. There was one single exception from this trend: For $log_{10}(C) < -3$, we observed that the accuracy for the training data decreased from 90% to 88%, while there was a numerical increase of the accuracy from 50% to 62.5% for the test data set. However, the increase of the classification accuracy to 62.5% at $C = 0.001$ was not significantly, as it fell within the 95%CI of the classification accuracy of 50% at $C = 1.0$ (95%CI [29.65, 70.35]). Note, that the parameter setting associated with a numerical increase in prediction accuracy could not be predicted on the basis of the training data, since the accuracy on training data was for $C = 0.001$ (88%) lower than for $C = 1$ (90%).

6.4 Discussion

In this Chapter, we presented a framework combining data mining methods to extract AD-typical patterns of brain atrophy using three different classifiers. We classified AD vs. HC, MCI-AD vs. MCI-MCI, and MCI vs. HC with excellent accuracy between 92% and 97.62% based upon leave-one-out validation. For the prediction of conversion from MCI to AD, the best predictive value was achieved with the VFI classifier reaching a predictive accuracy of about 75% validated in a train-and-test setting. As a proof of concept we first established the AD-specific spatial pattern of atrophy using

classifiers for the discrimination between AD and HC. Those brain regions that best discriminated AD from elderly HC included the medial temporal lobe, anterior cingulate gyrus extending towards the orbitofrontal cortex as well as the subcortical thalamic-basal ganglia brain areas, which were reliably identified using leave-one-out cross validation. These results are largely consistent with a previous PCA-based analysis [162], providing support for the convergent validity across different analysis methods. The pattern of atrophy we detected, agrees with findings of a range of previous independent MRI-based [116, 151] and neuropathological studies showing AD-typical predilection sites of pathological changes [143, 144]. For the classification of MCI-AD vs. MCI-MCI we obtained excellent results with a classification accuracy ranging from 91.97% to 95.83% with leave-one-out cross-validation. The skyline clusters are roughly a subset of the clusters identified for the separation of AD vs. HC, extending towards the temporal lobe including the superior temporal gyrus. Using AD and HC as training data and MCI as test data, we achieved an accuracy of 50% - 75% to predict conversion into AD. As expected, the performance of all classifiers declines in comparison to leave-one-out-cross-validation, since the test data originates from a data set that is expected to differ in the extent of pathological brain changes from the training data.

In our experiments, Bayes and VFI yielded superior results compared to the SVM approach. The leave-one-out experiments showed that the most selective regions for the discrimination of MCI-AD and MCI-MCI are roughly a subset of the most selective regions for AD vs. HC. Consequently, the training data contains many superfluous features, i.e. regions which are not selective to distinguish MCI-AD from MCI-MCI. VFI performs best on this difficult classification task, probably because this classifier is by design most robust w.r.t. superfluous information. The votes of these features approximately sum up to zero and thus have only minor effect on the classification accuracy. There is much more chance that random variations of the superflu-

ous features cause overfitting in SVM. We detected structural changes within the anterior cingulate gyrus, the hippocampus and the basal ganglia in MCI converters, consistent with the previously found pattern of atrophy in MCI compared to HC [135]. The major brain regions that separated best between HC and AD included primarily the prefrontal cortex especially the inferior and middle frontal gyri, the hippocampal region and adjacent subcortical basal ganglia, as well as more posterior brain regions within the parietal lobe. The hippocampal, frontal and parietal brain regions are well documented to be affected in AD. We detected also significant changes within the subcortical brain regions of AD.

Furthermore, the findings on the gray matter changes within the basal ganglia are consistent with previous results in the same patients data with a PCA based approach where the component that separated best between AD and HC was strongly associated with reduced volume of subcortical brain areas including the thalamus and caudate nucleus [162]. Thus, there is considerable overlap between different methodological approaches with regard to the detection of brain regions altered in an AD specific way. Previous independent studies have shown that considerable atrophic progression is found in subcallosal basal ganglia brain structures [61, 91, 92], and was demonstrated to show one of the fastest atrophy rates within the brain of AD patients (>15% per year, [164]). Although there is strong evidence of atrophy within these brain regions in AD, less attention may have been spent on the basal ganglia, since the cognitive function of these brain areas is not well known. A recent study, however, that detected strong atrophy within the thalamus and putamen of AD patients showed an association with global cognitive performance and executive functions independently from hippocampus grey matter atrophy [42]. Thus, the basal ganglia structures show pronounced volume reductions, consistent with our findings.

We aimed to render our analysis especially sensitive towards the detection of subtle brain abnormalities by employing a non-linear supervised fea-

ture selection method that is less dependent upon sample-size restrained power due to cross-validation and train-and-test than previous analysis for dimensionality reduction [40, 57, 161]. As an alternative to feature selection, dimensionality reduction can be achieved for example by principal component analysis and subsequently rated for class separation, using MANCOVA [161, 162, 163]. However, these methods depend entirely upon data-driven transformations and thus do not reduce variability in an informed way. There exist few supervised versions of singular value decomposition (SVD) and independent component analysis (ICA, e.g. [10, 149]) which consider the class labels during feature transformation. However, the results of these methods are difficult to interpret, since the amount of supervision is typically controlled by parameter settings. In contrast, the result of supervised feature selection is very intuitive because the interesting voxels are selected in the original image space. We decided to use the Information Gain as feature selection criterion, because it provides a very general rating of the discriminatory power, is highly efficient to compute and has been successfully applied in a large variety of applications, e. g. in information retrieval [121], object recognition [37] and bioinformatics [159, 140]. Correlation-based feature selection criteria, e.g. based on Pearson correlation, are closely related; however, the Information Gain is not restricted to linear correlations but captures any form of dependency between features and class labels. The applied feature selection technique can generally be used with a wide variety of classifiers. We selected three classifiers which represent different algorithmic paradigms and therefore provide a comprehensive evaluation of the discriminatory power of the selected features.

The majority of previous studies described characteristically altered brain areas in AD or MCI on a group-level [5, 24, 27, 41]. However, the diagnostic value of group-level analysis is limited. Some studies used multivariate methods which provide the potential to draw conclusions on a single-subject level but these papers do not report validated classification results [30]. Recent

studies reported validated classification results for the identification of AD. Duchesne and colleagues proposed to apply a support vector machine classifier based on least squares optimization on a selected volume of interest consisting of Jacobian determinants resulting from spatial normalization within the temporal lobe [52]. In contrast, we used the whole images of the subjects as single source for feature selection and classification. Fan *et al.* [58] present an approach for identification of schizophrenia relying on deformation-based morphometry and machine learning. They achieve high classification accuracy (91.8% for female subjects and 90.8% for male subjects). This approach is conceptually similar to ours since it also applies feature selection and watershed segmentation which can be regarded as some kind of clustering, before performing classification with SVM. For each voxel, a score is computed by linearly combining the discriminatory power for classification as measured by Pearson-moment correlation with the aspect of spatial consistency which is measured by intra-class correlation. Using a similar approach, Davatzikos *et al.* [40] report an accuracy of 90% for the identification of MCI in a leave-one-out validation setting. Our approach also emphasizes both aspects, the discriminatory power and the spatial coherency. However, very different definitions are applied to formalize these concepts. The discriminatory power is defined by the Information Gain, with the benefit to allow for arbitrary and not only linear correlations with the class label. Spatial coherency is achieved by density-based clustering which refines the selected features to form coherent regions. The result of watershed segmentation strongly depends on suitable selection of the thresholds which is very difficult especially in the presence of noise [69]. By the application of a modified density-based clustering technique our approach allows identifying the best discriminating brain regions without requiring any parameter settings or thresholds which are difficult to estimate.

It should be noted, that our study is based on a limited number of patients. In order to validate the utility of the classifiers further application to a

larger multicenter data set is necessary. In smaller samples the variability of the classification accuracy based upon the classifiers may be larger and thus less reliable (see [66] correspondence to [96]). The robustness of the results and potential influence of outliers has been tested by the leave-one-out validation, but may still need further testing in larger data sets. Another caveat of the current study is that the HC group was younger when compared to the MCI group. This age difference may have influenced the results. However, in a previous study it was shown in the same data set that age and gender were not significant predictors to group separation based upon a PCA scores [162]. Furthermore, the focus was on distinguishing between MCI converters and non-converters who did not differ in age in the current study.

Concerning the parameter settings for classification, for SVM there are two parameter choices which may have an impact on the classification result, the choice of the kernel and the choice of the complexity constant C. Due to the high dimensionality of the solution space (i.e. the high number of variables) it is indicated to use a linear kernel. Other, more complex kernel functions (such as polynomial, Gaussian, radial basis, etc.) are known to be subject to overfitting effects in presence of very high-dimensional spaces, i.e. good separation of the training data but deteriorated accuracy of the final classification result after validation. Therefore, more complex kernels should be used only if the training data is not well separable using the linear kernel. The results of the second tunable parameter of the SVM, the complexity constant C, show that varying of this parameter in a wide range did not lead to any significant differences in the classification accuracy with either the leave one-out paradigm or within the training-test validation scheme. This is probably due to the fact that in all our experiments the training data is sufficiently separable by a SVM using a linear kernel. In this case, the trade-off between margin maximization and training error minimization is of minor relevance in the optimization problem solved by SVM. Our results are consistent with those of LaConte *et al.* [103] who observed that the parameter

C has no influence on SVM as applied to fMRI data, unless the C value is very small ($C = 0.001$) [103]. Vemuri *et al.* [168] observed some influence of the parameter C on the classification result. For the classification of AD vs. HC based on MRI data only, best results with 85.8% in accuracy have been obtained using $C = 0.01$. With our method we achieved a classification accuracy of 90% for this task, independent of the selection of the parameter C between 0.001 and 1,000,000. Let us note that these findings are not directly comparable for several reasons: First, the study of Vemuri *et al.* is based on a larger collective involving 140 subjects with AD and 140 healthy controls and a different validation strategy has been applied. In our study, we applied leave-one-out cross-validation whereas Vemuri *et al.* applied four-fold cross validation.

In conclusion, we showed a novel approach to identify regions of high discriminatory power for the identification of AD and the prediction of conversion to AD among MCI. Our method combines data mining techniques from feature selection, clustering and classification and provides a concise visualization of the most selective regions in the original native image space.

Chapter 7

JGrid/FCC: Efficient Knowledge Extraction from MRI

Magnetic Resonance Imaging (MRI) allows to display different kinds of soft tissue with highest resolution and has attracted increased interest in the analysis of anatomical differences between normal and pathologic populations even in the field of neuroscience. Therefore, in the last Chapter, we have proposed the FCC framework combining several data mining techniques in order to provide a concise classification of pathogenous areas in the brain. However, to get a deeper insight into complex neurological abnormalities like dementia or somatoform disorder large-scale analysis is indispensable but mostly difficult and time consuming. As the FCC framework is only applicable to a small data set, in this Chapter, we propose JGrid, a highly efficient distributed computing system that allows a distributed computation of the FCC framework on a cluster of computers. Conforming to a standardized API provided by JGrid arbitrary software systems can be executed in a distributed fashion. After an introduction in the next Section, Section 7.2 describes the architecture of JGrid and the integration of FCC into an embedded system JGrid/FCC. The experiments (cf. Section 7.3) demonstrate the reduce in time usage distributing the computation to several client machines.

7.1 Introduction

Magnetic Resonance Imaging (MRI) has become an increasingly important role in understanding brain structure and analysis of anatomical differences between normal and pathologic populations. Despite the increasing interest in MRI studies, large-scale analysis of these images remains challenging as the processing and subsequent analysis of these high-dimensional MR images is very time consuming.

In Chapter 6, the framework FCC was presented that combines techniques from the field of data mining and machine learning to find patterns that are characteristic for a certain disease. In the first step, feature selection was applied followed by a density-based clustering algorithm on the reduced feature set to guarantee an examination of affected adjacent features in the brain. This clustering method produces clusters of features according to their spatial location and discriminatory power. In the second step, cross-validated classifiers derive a minimal set of features that enables the user for an optimized prediction of diagnosis and discrimination between diseased and healthy subjects. However, this framework is limited in its efficiency and therefore only applicable to small data sets.

In order to decrease the time consumption for the entire analysis a distributed computing system is required to use the capacity of several workstations. The most well-known software package for distributed computing is BOINC, the Berkley Open Infrastructure for Network Computing [3]. BOINC is a quasi-supercomputer for public-resource distributed computing that distributes the computing power to personal computers all over the world. With currently over 6,172,239 hosts and a processing rate of over 5,529 TeraFLOPS[1] BOINC outperforms the largest conventional supercom-

[1] `http://boincstats.com/`, Retrieved 03/15/2011

puter, the Tianhe-1A, that provides about 2,566 TeraFLOPs[1].

Oracle Grid Engine (OGE)[2], another well-known and widely used distributed computing software, is typically used on a computer farm or high-performance computing (HPC) cluster and supports automated load-balancing and distributed execution of large numbers of standalone, parallel or interactive user jobs on the best-suited machines in the cluster. Like BOINC, OGE supports the usage of very large clusters and provides features like shadow masters that take over in case of a crash of the controlling machines.

In this Chapter, we present JGrid that is implemented as a client/server system to manage a distributed computation of software that conforms to the API provided by JGrid. In contrast to most existing methods for distributed computing, JGrid is designed to handle small clusters of up to 30 nodes, has no permanently running daemon processes and no built-in features for remotely starting arbitrary processes or a dedicated command-line interface. Instead, it is tightly integrated into the Java ecosystem and controlled from within Java code using a specialized API or via a built-in graphical user interface. Furthermore, instead of remotely starting static programs, actual live Java Runnable objects are sent to and executed within special client processes on the remote machines.

By integrating FCC into JGrid to an embedded system JGrid/FCC, we provide a system that facilitates the execution of the whole framework on a cluster of computers and therefore reduces the computing time by magnitudes. We demonstrate in the experiments, that using JGrid/FCC on a set of 181 MRI scans reduces the computation time by 80% on a cluster of 10 client machines.

[1] `http://www.top500.org/list/2010/11/100`, latest TOP500 List November 2010, Retrieved 03/15/2011

[2] `http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html`. Retrieved 03/15/2011

7.2 JGrid/FCC

In this Section, we present JGrid, a solution for the distributed computation of the FCC framework presented in Chapter 6 including feature selection, clustering and classification in order to allow for a user friendly large scale analysis of MR images. We have implemented JGrid in Java and tailored it toward easy integration into existing software. This goal was realized mainly by relying on a Java API designed to integrate neatly into the Java ecosystem.

The underlying concept of JGrid is based on serializing Runnable objects. The interface `Runnable`, used in Java to encapsulate small bits of logic to be executed at a later time, was used to run the application logic on individual nodes of the computing cluster. JGrid adds on facilities to serialize these Runnable objects, distribute them over a network to a set of computers and execute the application logic on these; all of these features are accessible through a Java API.

Several functions like automatic classpath management, collecting output and error messages from the clients, functionality to coordinate access to load-sensitive resources and an API to load external JAR files into the remote JVM on the client side are implemented to obey to a set of standardized interfaces. This enables developers to easily add additional functionality. In this way, JGrid can be used to execute arbitrary software packages conforming to its built-in API.

A graphical user interface (GUI) allows for an easy handling of the software and the potential to easily integrate additional custom options and display custom information. External software that implements the provided interfaces can easily be loaded and executed by JGrid via the GUI.

In the following, we first provide an overview of the fundamental classes of JGrid and then describe the embedding of FCC into an integrated system JGrid/FCC.

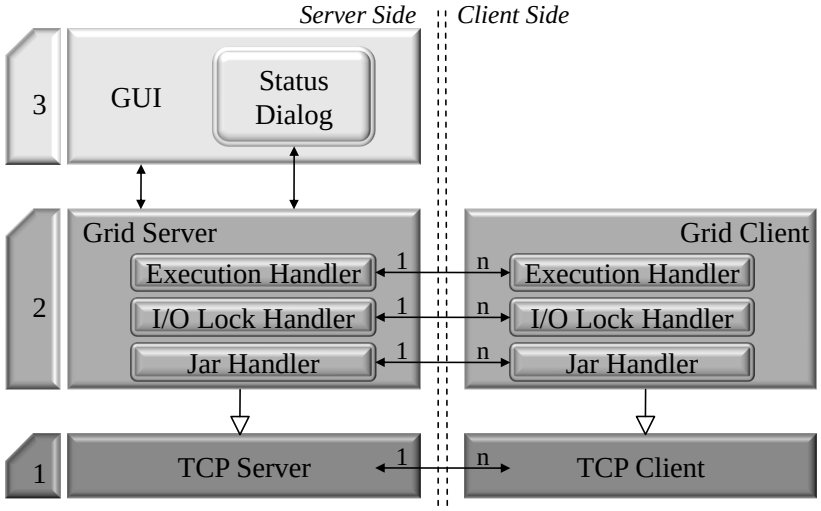


Figure 7.1: Distributed architecture of JGrid

7.2.1 Architecture

The JGrid software is organized in three distinct layers (cf. Figure 7.1). Each of them was designed to be functional without any of the other layers.

First Layer. This layer is comprised of the `TCPServer` and `TCPClient` components. These contain the logic for basic TCP/IP connection, exchange of serialized Java objects over the network and related functionality. `TCPServer` implements the server part of the first layer. It contains a Server Socket that listens for connecting clients. `TCPClient` implements the client side of the network. It supports connecting to a `TCPServer` instance over a network and receiving incoming serialized objects.

Second Layer. This layer contains the primary application logic of JGrid. It contains two components, `GridServer` and `GridClient`, which extend the afore-mentioned `TCPServer` and `TCPClient` components and add on facilities to run custom workflows on the grid, manage the execution of Runnable

objects on remote machines, synchronize the client nodes classpaths with the server and pool log and error messages on the server. In case external libraries are needed on the client machines, the `JarHandler` distributes the Jar files among the clients and inserts them to the clients' classpath at run time. A networked file system (NFS) is used in order to share data among clients. However, a NFS is not designed to handle large amount of simultaneous file I/O and does not support file locking. To avoid overloading the networked file system the `I/O LockHandler` ensures that no more than a given number of file system I/O operations are running at any given time. Executing Runnable objects on remote machines and keeping track of their state is done by the `ExecutionHandler` module, which also has both a server and client version. These communicate with each other using the facilities provided by the first layer to request and send Runnable objects, execute them and report status and result to the server.

Third Layer. This topmost layer adds on functionality to configure and run JGrid using a graphical user interface (GUI), shown in Figure 7.3. It should be noted that JGrid can also be used without this layer and controlled solely using its API. A `StatusDialog` is provided that can be used both standalone and as part of the third layer. It offers a straightforward display of the state and output of all clients.

The GUI consists of three important dialogs: A `LaunchPad` dialog used for configuration of the application, a `StatusDialog` to provide information on connected clients and running tasks and finally, an application-specific `ResultsDialog` used to present the results.

There are several ways to extend and customize the GUI: The launch pad dialog offers API functions to extend it with custom application specific settings. The status dialog allows for a custom progress section to be integrated and the results dialog content is completely customizable.

Another important functionality of the GUI layer is the ability to handle

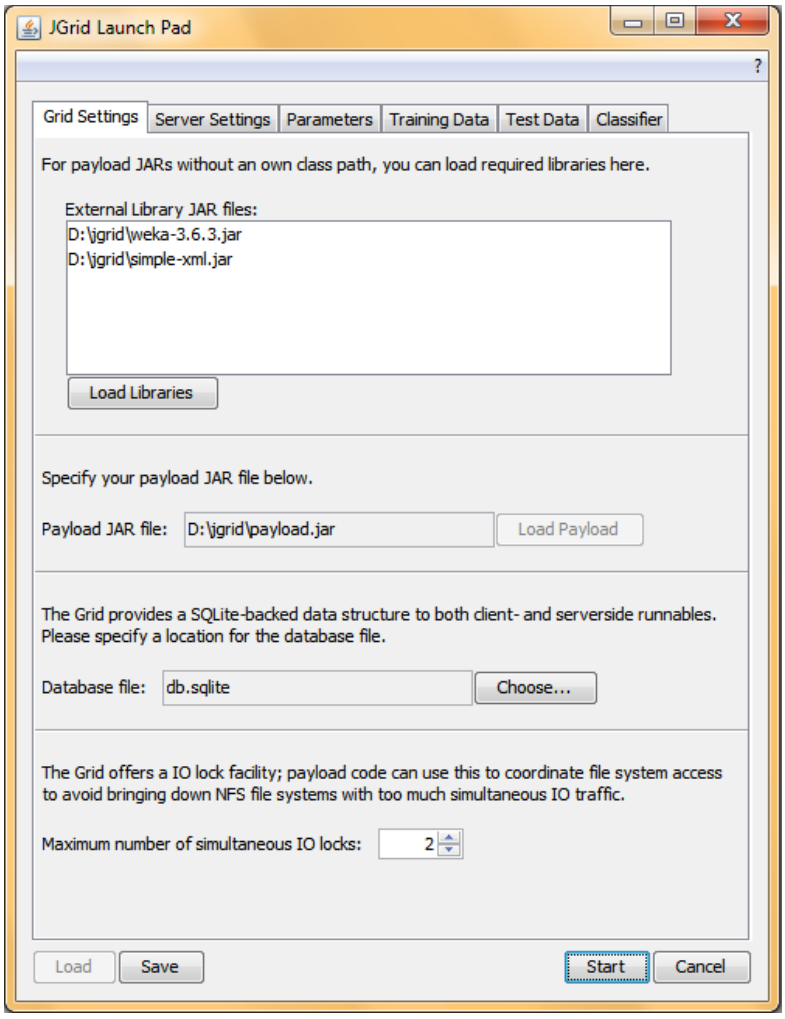


Figure 7.2: Graphical User Interface.

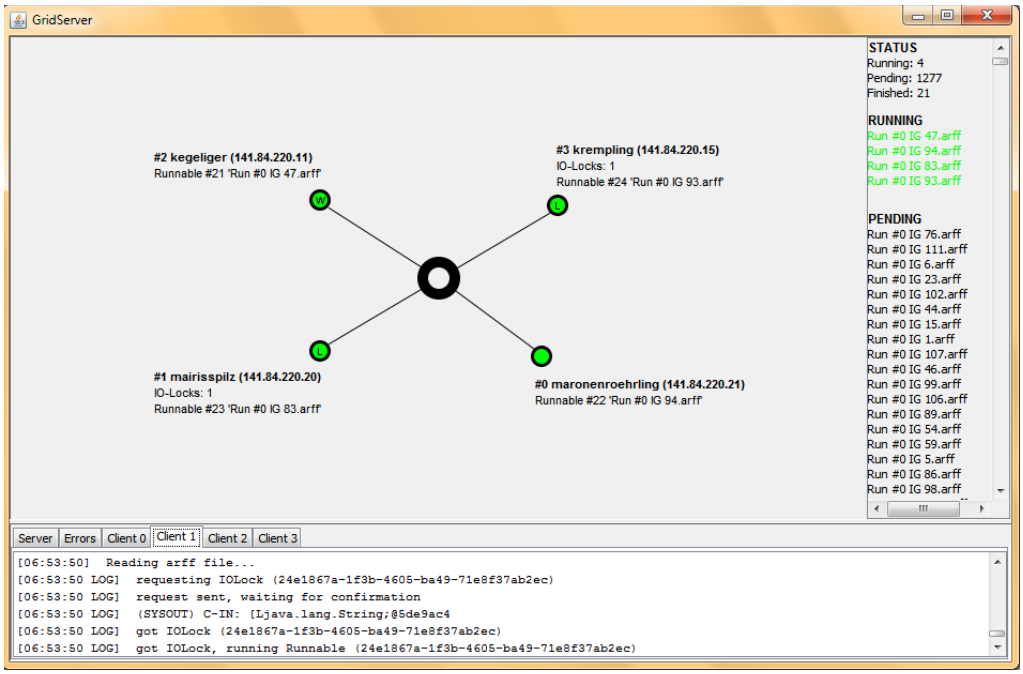


Figure 7.3: Status Dialog.

payload JAR archives. These are JAR files containing one or several classes conforming to the `Payload` interface provided by JGrid's API. This allows users to build self-contained application packages that can be loaded and used completely via the JGrid GUI and to make use of the convenience to distribute their own software to a cluster of computers.

7.2.2 The FCC Framework

The FCC framework is a multi-step data mining approach including feature selection, clustering and classification in order to find patterns in a set of MR images that are characteristic for a certain pathology. In the following, we provide a short description of the single steps. For a more detailed description, please refer to Chapter 6.

Feature Selection. MR images are composed of several million voxels that display the signal intensity of the contents of the corresponding volume element or voxel of the object being imaged. In order to fit into the available memory the data has to be split in several bins which can then be processed separately. The choice of the bin size has an effect on the runtime, as demonstrated in the experiments. Furthermore, as only a subset of the voxels carry information for class separation, feature selection has to be applied to reduce the amount of voxels and to select the most discriminating features. The Information Gain (IG) [146] rates the interestingness of a voxel for class separation. Let DS be a set of MRI scans each being labeled to a set of discrete classes $C = \{c_1, \cdots, c_k\}$, where each class corresponds to a certain pathology. The IG of a voxel v_i is defined by the amount by which the entropy of the class distribution $H(C)$ decreases through the additional information provided by v_i on the class, which is described by the conditional entropy $H(C|v_i)$ [141].

$$IG(v_i) = H(C) - H(C|v_i)$$

Clustering. After feature selection, clustering is applied on the whole reduced feature set to find groups of adjacent voxels in the scans that have high discriminatory power. Therefore, the density-based clustering approach DBSCAN [55] has been modified for clustering voxels. The definition for core object property and direct density reachability has been adjusted as follows: A voxel v_i is called a *core voxel* if the IG of the voxel is larger than a predefined threshold t and is surrounded by at least *MinVox* voxels also having an IG at least t.

Feature Classification. After clustering, several classification algorithms are applied to validate the discriminatory power of the selected features. These classification algorithms include Linear Support Vector Machine [142], Bayesian Classifier [89] and Voting Feature Intervals [47]. To validate the

found discriminatory patterns, two different validation techniques, train-and-test and cross-validation, are applied.

Result. As a result the spatial location of the high discriminating clusters are obtained together with a quantitative validation. Furthermore, several statistics concerning the number of clusters as well as the size and the IG of the clusters are provided. This enables the user for a concise interpretation of the results as well as an effective diagnosis.

The whole FCC framework was implemented to conform to the payload interface and packed into a JAR-file. Loading this JAR-file by the JGrid GUI the integrated JGrid/FCC system enables the user for an distributed computation on a cluster of computers. For this purpose, the whole workflow is split into parallelizable Runnables objects which are organized in a data structure that manages the dependencies between the different Runnables. This ensures the compliance of the workflow and a correct execution of the Runnables in correct order. External libraries are automatically loaded and distributed to the clients machines. Several user interaction including data selection and parameter settings as e.g. number of fold for the cross-validation scheme can be done using the graphical user interface. However, distributing the workflow to a cluster of computers does not influence the comprehensive power of the FCC framework.

7.3 Experiments

We evaluated JGrid/FCC on a set of 181 MRI scans obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI)[1]. MRI examinations were performed on a 1.5 Tesla MRI scanner. This set comprises images of patients that suffer from Mild Cognitive Impairment (MCI) that underwent

[1] `http://adni.loni.ucla.edu/`

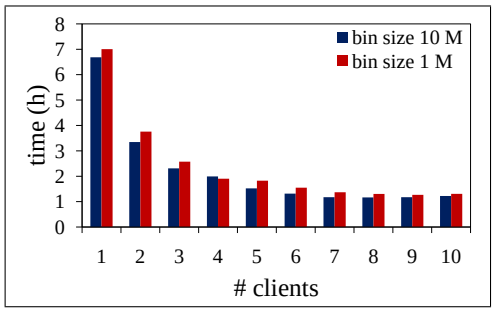


Figure 7.4: Runtime of the whole workflow vs. number of clients.

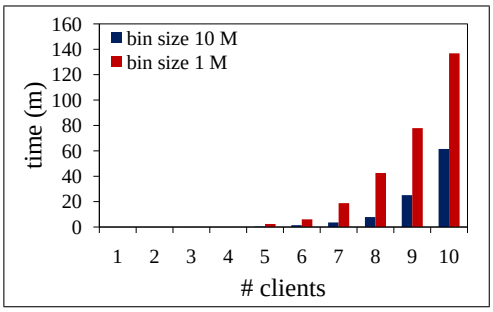


Figure 7.5: I/O lock and waiting time vs. number of clients.

a five year follow-up study. All images were preprocessed using the statistical software package SMP8[1]. In order to test the performance of JGrid/FCC we applied a tenfold cross-validation in each experiment. All tests were performed on workstations equipped with AMD Phenom(tm) II X4 B95 Processors and 8 GByte of RAM on each machine. The MR images are stored in a public NFS. To protect it from overload we allowed for a maximum of four concurrent I/O locks. All tests were conducted for two different bin size values of one and ten million.

Figure 7.4 shows the runtime for JGrid/FCC with different number of clients. It can be seen that the bin size has no significant impact on the performance even though a bin size of 10 million leads to slightly better results. Greatest decrease of overall runtime can be achieved with eight clients and bin size set to ten million reducing the runtime to process the whole workflow by 82% from 6.68 hours to 1.16 hours. Increasing the number of clusters to nine or ten results in marginally worse results. This trend correlates with a dramatic increase in time spent waiting for I/O locks to be granted as can be seen in Figure 7.5.

This diagram illustrates the correlation between the number of clients,

[1] `http://www.fil.ion.ucl.ac.uk/spm/software/spm8/`

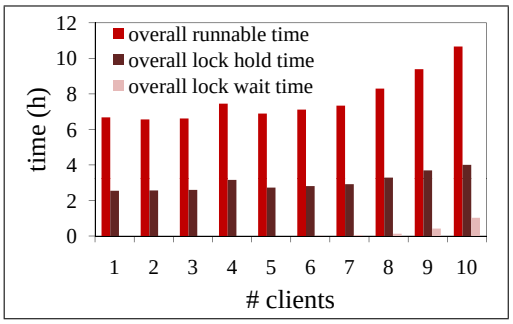


Figure 7.6: Runtime in different states of the workflow vs. number of clients.

the bin size and the overall time spent waiting for I/O locks to be granted. The waiting time increases drastically with higher number of clients. Apart from that, here the choice of the bin size has a great effect on the waiting time, too, with a smaller value causing a large decrease in speed. A bin size of one million leads to significantly worse performance compared to a value of ten million. This is due to the greater number of individual Runnables to which the same amount of data is assigned, resulting in shorter runtime per Runnable and a larger number of I/O locks that need to be requested and granted. There is also a small overhead in the I/O lock mechanism due to requesting, granting and revoking locks and sums up to higher waiting time for many short-duration I/O locks then with only a few high-volume file system operations.

The time waiting for I/O locks to be granted has an effect on the overall runtime. This can be seen when measuring the time spent executing the Runnables summed up over all clients. Figure 7.6 illustrates the time needed for executing the Runnables, holding and waiting for the I/O locks with a bin size of ten million summed up over all clients, respectively. The time spent holding the I/O locks remains relatively constant even when increasing the number of clients. Marginal fluctuation in I/O lock hold time is due

to varying load of the NFS resulting in varying time-use for read and write operations. The time spent for waiting for an I/O lock confirms to the values charted in Figure 7.5. As already mentioned, a higher number of clients causes an increased I/O lock waiting time. Apart from this, a consistent increase in overall Runnable execution time is visible, which includes both I/O wait and hold time, since I/O locks are requested and used within Runnables exclusively. The increase in I/O wait time with additional clients and worse NFS performance mostly explain the Runnable execution time increase.
However, while the result of JGrid/FCC is guaranteed to be equivalent to that of FCC, we demonstrate a high speed-up distributing the application logic to a cluster of computers.

Chapter 8

Motif Discovery in Brain Networks of Patients with Somatoform Pain Disorder

In the last Chapter we have considered single voxels in MRI scans for analysis but did not consider the relationship among them. In this Chapter, we go one step ahead, and model the image data as a graph, where nodes represent single voxels and edges connections between voxels that share same or similar activation of the underlying brain compartment. More precisely, we applied different graph mining algorithms to brain co-activation networks of patients with somatoform pain disorder and healthy controls in order to find frequent subgraphs in the brain that are characteristic for the disease. After a general introduction, Section 8.2 surveys related work in the area of frequent subgraph mining. Section 8.3 introduces basic definitions from graph theory. Details about network construction and evaluation of detected motifs are described in Section 8.4. We applied the heuristic approach GREW and the exhaustive approach FANMOD to these networks. The results are presented in Section 8.5. Basic ideas of this Chapter have been published in [130].

8.1 Introduction

Graph mining or network analysis has become increasingly important in the last decade, even in the area of Web 2.0 [129] which is associated with web applications that facilitate user interaction in form of social networks, blogs, etc. [45]. The analysis of these networks is both of scientific and commercial interest. On the one hand, psychologists want to study the complex social dynamics between individuals, and industry wants to analyze these networks for marketing purposes. Detecting influential individuals in a group of people, often referred to as 'key-players' or 'trend-setters', is relevant for marketing, as companies could then focus their advertising efforts on persons known to influence the behavior of a larger group of people.

In the field of bioinformatics, the influence of chemical compounds on physiological processes, e.g. toxicity, protein protein interaction (PPI) networks or regulatory networks are studied extensively. Furthermore, characterization of these complex structures can be accomplished through the discovery of basic substructures that are frequently occurring. Identification of such repeating patterns might be useful for diverse biological applications such as classification of protein structural families, investigation of large and frequent sub-pathways in metabolic networks, and decomposition of PPI networks into motifs. Motifs are substructures that appear in at least t of the graphs in the data set.

In this Chapter, we focus on the mechanisms in the brain that are associated with somatoform pain disorder. Somatization disorders constitute a large, clinically important health care problem that urgently needs deeper insight [173]. Earlier studies have revealed that different subunits within the human brain interact among each other, when particular stimuli are transmitted to the brain. Hence, the different components form a network. From an algorithmic point of view, interacting subunits act as specific subgraphs

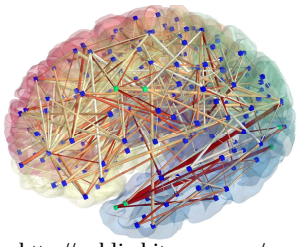


Figure 8.1: Brain network.

within the network. From a medical perspective, it is interesting to ask to which degree interactions of subunits are correlated with medical disorders.

We give an answer to the question whether brain compartments of patients with somatoform pain disorder form motifs that differ from brain compartments of healthy controls. For this purpose, we analyze task-fMRI scans of the brain of ten subjects, six patients with somatoform pain disorder and four healthy controls that attended the study of [74]. In this study both groups underwent alternate phases of non-pain and pain stimuli during the fMRI scanning. We construct a brain co-activation network for each subject where each node represents a voxel in the fMRI image (cf. Figure 8.1). Voxels are grouped together in 90 so-called regions of interest (ROIs) using the template of [166]. To uncover frequent subgraphs in each of these networks, we apply two approaches, the efficient heuristic approach GREW [102] and the exhaustive sampling technique FANMOD by [172]. While an heuristic approach allows for finding motifs of arbitrary size, an exhaustive sampling algorithm can be applied to find *all* frequent subgraphs, but in this case we are limited in the size of the subgraphs. Both algorithms are designed to operate on one large graph and to find patterns corresponding to connected subgraphs that have a large number of vertex-disjoint embeddings. We demonstrate that patients with somatoform disorder show activation patterns in the brain that are different from those of healthy subjects.

8.2 Related work

Several algorithms have been defined for finding frequent subgraphs and their embeddings in one large graph or in a data set of graphs. We distinguish algorithms for 'graph data set mining' that work on a data set of graphs, and algorithms for 'large graph mining' that discover frequent motifs in one large graph. While frequent subgraph algorithms that work on large graphs can directly be applied to data sets of graphs, the other direction is more complicated. However, by splitting a large graph into subgraphs, one can still use a graph data set mining algorithm for frequent subgraph discovery on the large graph (albeit some subgraphs might be lost by the splitting).

8.2.1 Graph Data Set Mining

For the graph data set mining task, approaches can be broadly divided into two classes, apriori-based approaches and pattern-growth based approaches. AGM (Apriori-based Graph Mining)[86] determines subgraphs G' in a data set DS of graphs that occur in at least *minsup* percentage of all graphs in the data set. AGM works on graphs with edge and node labels. In principle, AGM uses the famous apriori [1] principle of iterated candidate generation and candidate evaluation. Candidate generation means that candidate subgraphs are created by joining subgraphs that have been shown to be frequent in earlier iterations. In the candidate evaluation phase, these candidates are tested, i.e. it is checked whether their frequency is greater than *minsup*, and then the whole process is iterated until all frequent patterns have been found. AGM uses a canonical form and a normal form to represent subgraphs to reduce runtime cost for subgraph isomorphism checking.

Similar to AGM, FSG (Frequent SubGraph Discovery) [100] uses a canonical labeling based on the adjacency matrix. Canonical labeling, candidate generation and evaluation are sped up in FSG by using graph invariants and

the Transaction ID principle, which stores the ID of transactions a subgraph appeared in. This speed-up is paid for by reducing the class of subgraphs discovered to connected subgraphs, i.e. subgraphs where a path exists between all pairs of nodes.

The most well-known member of the class of pattern-growth algorithms, gSpan (graph-based Substructure pattern mining), discovers frequent substructures efficiently without candidate generation [175]. Tree representations of graphs are encoded using a Depth First Search (DFS) code, amongst which a minimum DFS code is chosen according to some lexicographic order. Pre-order DFS-tree search is then conducted to find the complete set of frequent subgraphs in a set of graphs. gSpan is efficient, both w.r.t. runtime and memory requirements, making it one of the best state-of-the-art algorithms for graph data set mining.
CloseGraph [176] extends gSpan by limiting the search to frequent complete graphs, i.e. subgraphs without supergraphs that have the same support, thereby increasing the efficiency of mining substantially.

8.2.2 Large Graph Mining

Unlike graph data set mining, large graph mining intends to find subgraphs that have *minsup* embeddings in one large graph.

Milo *et al.* [118] was the first to define network motifs and find them. By exhaustively enumerating all subgraphs this method is limited to subgraphs with three or four nodes.

SUBDUE [36] tries to minimize the minimum description length (MDL) of a graph by compressing frequent subgraphs. Frequent subgraphs are replaced by one single node and the MDL of the remaining graph is then

determined. Those subgraphs whose compression minimizes the MDL are considered as frequent patterns in the input graph. The candidate graphs are generated starting from single nodes to subgraphs with several nodes, using a computationally-constrained beam search.

GREW [102] and SUBDUE are greedy heuristic approaches to frequent graph mining that trade speed for completeness of the solution. GREW iteratively joins frequent pairs of nodes into one super-node and determines disjoint embeddings of connected subgraphs by a maximal independent set algorithm. Similarly, vSIGRAM and hSIGRAM [101] find subgraphs that are frequently embedded within a large sparse graph, using "horizontal"(h) breadth-first search and "vertical" (v) depth-first search, respectively. They employ efficient algorithms for candidate generation and candidate evaluation that exploit the sparseness of the graph.

In [106], different sampling methods for a single large graph are compared. These algorithms sample subgraphs from a large network and estimate their frequency rather than resorting to heuristics.
Kashtan *et al.* [93] presented a sampling method for subgraph counting by picking a random edge from the network and then expanding the corresponding subgraph by successively picking neighboring random edges. In contrast to picking only one single edge, Zhou *et al.* [182] presented a method for frequent subgraph mining using the sampling method 'random areas selection sampling' that randomly selects one area of the large graph. FANMOD, the algorithm by [172] uses a randomized enumeration strategy for sampling subgraphs. If one does not want to resort to any heuristics and miss out on any embeddings of a frequent subgraph, one may employ this enumeration strategy for exhaustively enumerating all subgraphs rather than randomly sampling from this enumeration.

8.3 Basic Definitions

We start with a brief summary of necessary definitions from the field of graph mining.

Definition 10 (Labeled graph / network) *A* **labeled graph** *is represented by a 4-tuple $G = (V, E, L, l)$, where V is a set of vertices (i.e. nodes), $E \subseteq V \times V$ is a set of edges, L is a set of labels, and $l : V \cup E \rightarrow L$ is a mapping that assigns labels to vertices V and edges E. If labels are not of decisive importance, we will use the* short *definition of a graph $G = (V, E)$. In the following we also use* ***network*** *as a synonym for graph.*

Definition 11 (Subgraph) *Let $G_1 = (V_1, E_1, L_1, l_1)$ and $G_2 = (V_2, E_2, L_2, l_2)$ be labeled graphs. G_1 is a* **subgraph** *of G_2 ($G_1 \sqsubseteq G_2$) if the following conditions hold: $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, $L_1 \subseteq L_2$, $l_1 = l_2$. If G_1 is a subgraph of G_2, then G_2 contains G_1.*

Definition 12 (Isomorphism) *Two graphs are* **isomorphic** *if there exists a bijection f between the nodes of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that $(v_{1a}, v_{1b}) \in E_1$ iff $(v_{2a}, v_{2b}) \in E_2$ where $v_{2a} = f(v_{1a})$ and $v_{2b} = f(v_{1b})$. If G_1 is isomorphic to G_2, we will refer to (v_{1a}, v_{1b}) and (v_{2a}, v_{2b}) as* **corresponding edges** *in the following.*

The problem to decide whether two graphs are isomorphic, i.e. the *graph isomorphism problem*, is not yet known to be NP-complete or in P. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *subgraph isomorphism problem* consists in finding a subgraph of G_2 that is isomorphic to G_1. This problem is known to be NP-complete [67].

Definition 13 (Embedding) *If graph G_1 is isomorphic to a subgraph S of graph G_2, then S is referred to as an* **embedding** *of G_1 in G_2.*

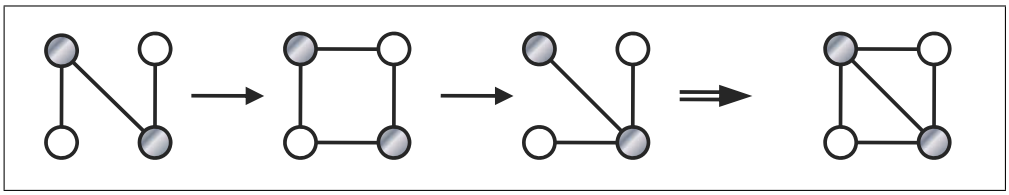


Figure 8.2: Transformation from a time series of three labeled graphs into the corresponding union graph.

Definition 14 (Frequent subgraph / motif) *A graph G_1 is a* **frequent subgraph** *of graph G_2 if G_2 contains at least t embeddings of G_1, where t is a user-set frequency threshold parameter. Such a frequent subgraph is often called a* **motif**.

Definition 15 (Union graph) *Given a time series of graphs G_{ts} with n states. Then the* **union graph** *$DG(G_{ts})$ of G_{ts} is defined as $DG(G_{ts}) = (V_{DG}, E_{DG}, \ell)$, where $V_{DG} = V_i$ for all $1 \leq i \leq n$ and $E_{DG} = \cup_{1 \leq i \leq n} E_i$.*

An example for the transformation of a time series of graphs into a union graph is depicted in Figure 8.2. Note that the union of all edges of the time series is the set of edges of the union graph.

8.4 Finding Motifs in a Brain Network

In this Section, we first describe how a time series of fMRI image data is converted into a labeled graph, and then we describe how frequent subgraph mining algorithms can be applied to these networks. At the end, we present the evaluation of the detected motifs.

8.4.1 Construction of Brain Co-Activation Networks Out of fMRI Time Series

As we are interested in topological patterns that are characteristic for patients with somatoform pain disorder, we perform graph mining on brain co-activation networks. Each detected motif represents interacting regions of the brain.

In order to receive such network models, fMRI time series data can be transformed in the following manner. We define the voxels of the fMRI image data as the vertex set V. The measured value of a voxel indicates the degree of blood circulation in the particular brain region. The *darker* the voxel, the *more* blood is present in the compartment, thus the *higher* the activation is. Edges between two vertices v_1 and v_2 stand for a similar level of activation of the two corresponding voxels. We distinguish two categories of activation levels for each voxel v at each time point i, denoted by v_i. $a(i)$ stands for its activation level at time point i. We determine an *activity–score* for each voxel v_i by comparing the median activation level of v across the time series with $a(i)$ in order to assign activation categories.

$$activity\text{–}score(v_i) = \frac{a(i) - median_{j \in \{1,..,n\}} a(j)}{median_{k \in \{1,..,n\}} \left| (a(k) - median_{j \in \{1,..,n\}} a(j) \right|}$$

We use a median-based *activity–score* rather than a mean-based as we want to detect unusually high activation levels. In contrast to a mean-based *activity–score* a median-based *activity–score* is more robust with respect to these extremes and better suited for detecting them, as validated in initial experiments (not shown here).

- **high activation**: $a(i)$ is significantly higher than the median activation level of v.
  ($activity\text{–}score(v(i)) \geq 7.0$)

- **no significant activation**: $a(i)$ is not significantly higher than the median activation level of v.
  $(activation\text{–}score(v(i)) < 7.0)$.

Edges between vertices v_1 and v_2 are assigned if v_1 and v_2 both show high activation. Finally, we perform frequent subgraph mining on the resulting union graph.

8.4.2 Performing Frequent Subgraph Mining on Brain Co-Activation Networks

In order to find frequent subgraphs in our network, we have to group our nodes and assign each group a labeling. A meaningful grouping of nodes when considering brain networks is a mapping of the nodes to their corresponding brain compartments. Thus, motifs in those networks represent compartments of the brain that show a similar activation profile. We remove edges between nodes that share the same label, as a correlated degree of activation within one region is trivial, and we are interested in activity of different regions.

Then we apply two large graph mining algorithms for finding motifs in our labeled graphs. First, we employ the heuristic approach GREW [102], as the runtime effort for exhaustive enumeration grows exponentially in the size of the subgraphs. This enables us to find frequent subgraphs in a large graph rather than to restrict ourselves to small frequent subgraphs. Nevertheless, as we want to avoid missing out on embeddings of motifs that consist of a small number of vertices (up to six), we also employ the exhaustive enumeration strategy FANMOD by [172].

8.4.3 Evaluation of Detected Motifs

To find motifs that are characteristic for a disease, we have to analyze the motifs separately. Therefore, we want to detect motifs that occur in patients

but not in the control group and vice versa. Another class of motifs that might be interesting are motifs that occur in all subjects that attend a certain study.

Another aspect that should be considered is the label distribution across motifs. A label that is used for a large number of vertices inside the network model of a particular subject s has a higher probability to appear in a motif than a label that covers a small number of nodes. Hence, we have to define the normalized frequency of a node label l, denoted by $freq_{norm}(l)$.

$$freq_{norm}(l) = \sum_{s \in S} \frac{freq_{m \in \{1,\ldots,n\}}(l) \cdot \#Embeddings(m|s)}{freq_{Background}(l|s)}$$

$freq_{m \in \{1,\ldots,n\}}$ stands for the number of occurrences of label l in a motif m. This number has to be multiplied by the number of isomorphic subgraphs of m found in a subject s, its embeddings found in s. The $freq_{Background}(l|s)$ describes the number of occurrences of label l with respect to all vertices of the network that refers to subject s. In our case this is equivalent to the *size* of a ROI. Finally, we sum up over all subjects given in the data set.

8.5 Experiments

First, we summarize the construction of the network models, including a detailed description of the used data set and the labeling scheme for the vertices of the networks. Then we perform motif discovery using the heuristic algorithm GREW to allow for finding motifs of arbitrary size. As an evaluation of these results indicates that a multitude of the motifs found by GREW consist of only a small number of vertices, we go a step further and additionally apply the exhaustive sampling technique FANMOD by [172] on the network models. Finally, we compare the results of both algorithms and give representative motifs for the group of subjects that suffer from the somatoform pain disorder and typical motifs for the group of healthy controls.

Construction of the Brain Network Models. We created networks for 10 subjects that attended the studies of [74]. The resulting networks comprise 66 to 440 nodes with 90 different classes of node labels and 358 to 13,548 edges. In addition, the network models indicate different number of edge types. These refer to the concatenation of the labels of the adjacent nodes. All edges are undirected because in the relationship 'both adjacent voxels show high activation' a direction makes no sense. The exact statistics of each subject are depicted in Table 8.1.

Table 8.1: Statistics of the network models for each subject.

Subject	# Vertices	# Different Labels	# Edges	# Edge Types	# Different Motifs by GREW
Patient 1	102	31	453	91	38
Patient 2	185	32	1,961	84	706
Patient 3	241	35	3,506	90	505
Patient 4	263	46	5,152	293	752
Patient 5	313	46	10,977	372	4,154
Patient 6	440	58	13,548	475	4,256
Control 1	66	10	358	15	15
Control 2	99	33	407	111	32
Control 3	109	18	1,093	13	133
Control 4	202	35	2,045	133	236

Time Series Data Sets. We used fMRI time series data (1.5 T MR scanner) of six female somatoform patients and four healthy controls. Standard data preprocessing including realignment, correction for motion artifacts and normalization to standard space have been performed using SPM2 (available at `http://www.fil.ion.ucl.ac.uk/spm/`). In addition, to remove global effects the voxel time series have been corrected regressing out the global mean, as suggested in [150].

Vertex Labels. We labeled all nodes in our network model by regional parcellation of the voxels into 90 brain regions using the template of Tzourio-Mazoyer *et al.* [166].

Finding Motifs With GREW. To allow for finding motifs of arbitrary size, we searched for topological motifs using GREW with a frequency threshold of $t = 5$. In these experiments, we searched for motifs with a minimum number of one edge. The total number of different motifs found in the ten networks is depicted in the last column of Table 8.1.

Evaluation of the Motifs Detected by GREW. Altogether we found 10,530 different motifs in somatoform patients and healthy controls. 10,173 different motifs were detected among patients, 413 within the group of healthy subjects, where some of these motifs were also found among the patients and vice versa.

For validation, we divided the subjects into three classes. Class (1) contains only the somatoform patients, class (2) consists of the controls exclusively and class (3) composes the union of class (1) and (2). Figure 8.3 shows typical representatives of each class. The two motifs on the left occur in 57% of the patients but in no healthy subject. The middle motif arises in 50% of the class (2)–subjects but in no patient. The upper motif on the right-hand side was found in 50% of the control group and in 14% of the patients, the lower motif in 25% of the control group and in 43% of the patient group. Most of the typical representatives of both groups comprise only a small number of vertices.

The largest motifs (highest number of vertices and edges) of class (1) were found in subject 'patient 5'. They consist of 28 vertices and 29 edges, five different brain compartments are involved in this motif. A total of 34 motifs of this kind were found in this subject. The largest motifs in class

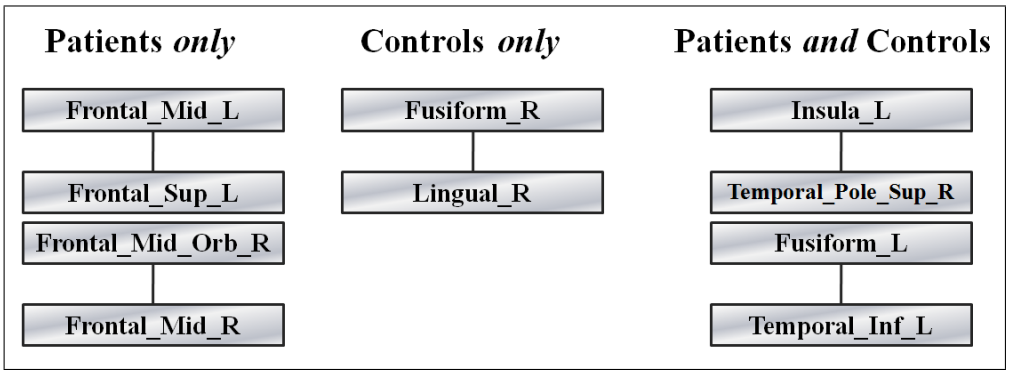


Figure 8.3: Typical representatives of motifs found in the groups of somatoform patients, healthy controls respectively and the group of all subjects.

(2) were detected in subject 'control 3'. We found two motifs that comprise 12 nodes with two different labels and 17 edges. An example of the largest motifs found in class (1) and the two largest motifs of class (2) are shown in Figure 8.4. Note that no motif occurs in all subjects.

Evaluation of ROIs. We found motifs that can discriminate well between somatoform patients and controls. In the next step, we determined the normalized frequencies of the ROIs in patients and controls, respectively. Figure 8.5 illustrates the results for the 15 most frequent ROIs w.r.t. the patient group, and Figure 8.6 for the group of healthy controls, respectively. Within the motifs found in the patient group, *Caudate_R* is the most frequent ROI ($freq_{norm} = 6.43\%$). The most frequent brain region w.r.t. the motifs of the control group is *Frontal_Mid_Orb_R* ($freq_{norm} = 22.04\%$). ROIs that show a small frequency are summarized by the label *Miscellaneous*. Our results are consistent with a previous study [74]. They report different activation pattern in the regions *Insula_L* (Patients: $freq_{norm} = 5.29\%$ Controls: $freq_{norm} = 1.95\%$) and *Frontal_Mid_Orb_R* (Patients: $freq_{norm} = 3.38\%$ Controls: $freq_{norm} = 22.04\%$), the key-player in the group of healthy controls. Our results of different activation of the parahippocampal cortex in

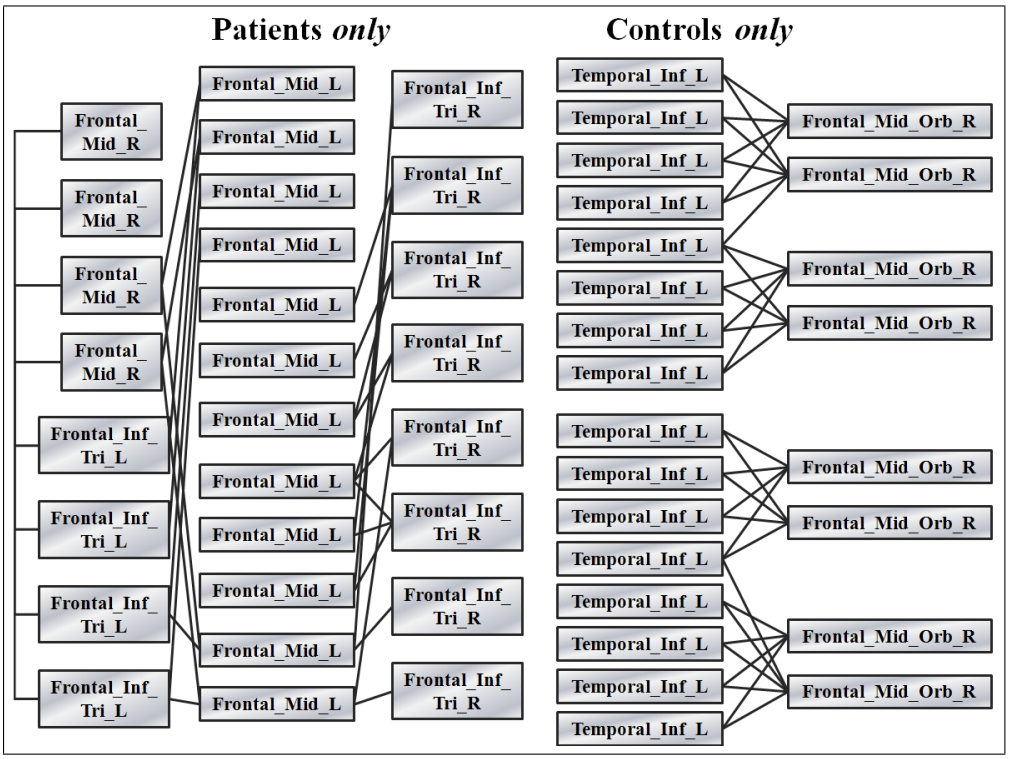


Figure 8.4: Largest motifs found in the groups of somatoform patients and healthy controls.

patients and controls (not depicted in Figures 8.5 and 8.6) supports a recent study that suggests that patients with posttraumatic stress disorder showed also an altered activation pattern in the parahippocampal cortex in comparison to healthy controls when subjected to painful heat stimuli [70]. In addition, we found that patients show increased activation in *Rolandic_Oper_L*, *Caudate_R* and *Rectus_R* whereas the control group is activated in the regions *Temporal_Inf_L*, *Heschl_R* and *Lingual_R* to a higher degree. Also, the olfactory region shows alterations in the activation of patients and controls. Whereas *Olfactory_R* occurs to a much higher degree in motifs found in patients, motifs found in the networks of controls are labeled more often with *Olfactory_L*.

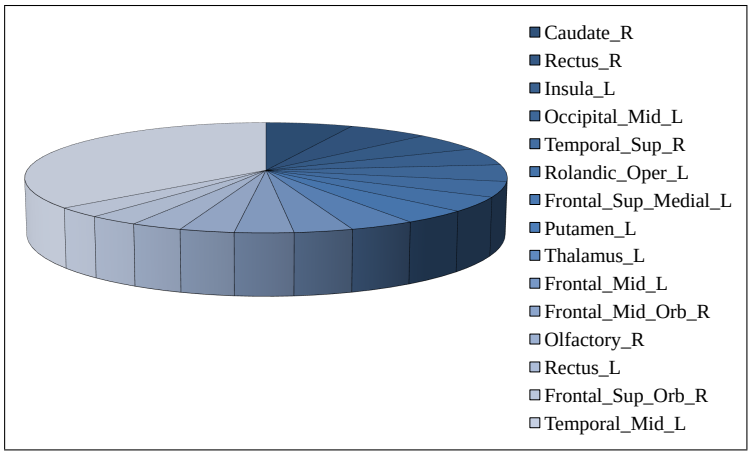


Figure 8.5: Frequencies of ROIs in motifs of patients.

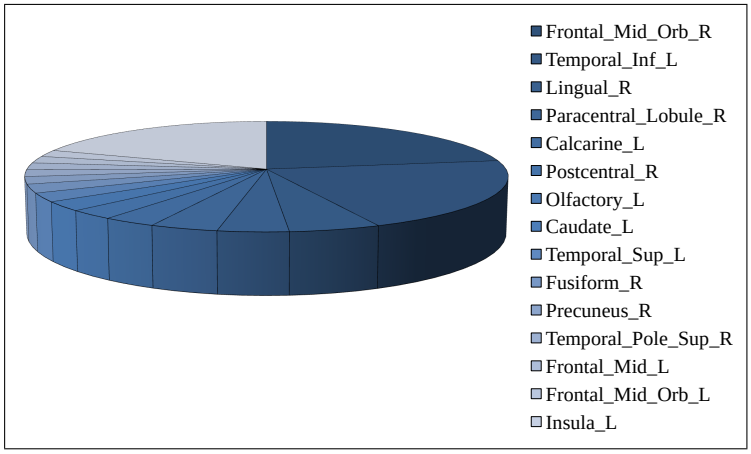


Figure 8.6: Frequencies of ROIs in motifs of controls.

Finding Motifs With FANMOD. The public disposable implementation[1] of the exhaustive sampling algorithm by [172] is restricted to networks with a maximum number of 16 different vertex labels. Hence, we map the original network data to a modified version where only the 15 most frequent ROIs are kept. The remaining vertices are labeled by *Miscellaneous*. This labeling scheme refers to the ROI distribution among motifs detected by GREW as shown in Figure 8.5 or 8.6 . As edges of the type *Miscellaneous*–*Miscellaneous* occur disproportional frequent within the network, and would therefore tamper with the real frequencies of the results, we deleted that kind of interaction.

We searched for motifs, consisting of up to six vertices and used a sample rate of 100,000 to estimate the number of subgraphs which is the default parameterization, as recommended by the authors. Table 8.2 illustrates the number of motifs that comprise $3 \leq k \leq 6$ vertices for each subject. We compared these results by applying GREW ($t = 5$) to the same modified networks. The last column of Table 8.2 demonstrates that for small subgraph sizes an exhaustive method provides significant more information about the graph structure. Note that the total number of detected motifs by FANMOD with a limited number of vertices exceeds the total number of motifs found by GREW with arbitrary size in most cases. However, an exhaustive method is only applicable for finding subgraphs that consist of a small number of vertices.

[1] `http://theinf1.informatik.uni-jena.de/motifs/`

Table 8.2: Motifs detected by FANMOD for different number of vertices compared to the number of motifs found by GREW under the same conditions. For some motif sizes, FANMOD is not applicable as the runtime effort for exhaustive enumeration grows exponentially in the size of the subgraphs.

Subject	# Motifs ($k = 3$)	# Motifs ($k = 4$)	# Motifs ($k = 5$)	# Motifs ($k = 6$)	# Motifs ($3 \leq k \leq 6$)	# Motifs by GREW
Patient 1	9	13	28	42	92	4
Patient 2	42	121	288	-	451	406
Patient 3	5	6	7	8	26	4
Patient 4	22	91	374	-	487	56
Patient 5	139	509	-	-	648	988
Patient 6	431	3,292	-	-	3,723	1,861
Control 1	6	10	16	22	54	34
Control 2	18	32	51	78	179	56
Control 3	9	26	-	-	35	161
Control 4	137	565	2,399	-	3,101	472

Part IV

Conclusions

Chapter 9

Summary and Future Directions

Data mining is the key part of the KDD process and is the application of algorithms to discover patterns in large databases. Clustering and Classification are among the most important data mining tasks. In this thesis, we focus on new clustering methods and application in biomedical databases. This Chapter summarizes the main contributions of this thesis in Section 9.1 and shows some directions for future work in Section 9.2.

9.1 Summary of Contributions

The rapidly increasing amount of data even in the field of biology or medicine, requires effective and efficient data mining methods to gain new information contained in the collected data. In this thesis, we developed new clustering methods and demonstrated the application of data mining methods like clustering, classification or graph mining to medical images in order to get deeper insight into neurological diseases like Alzheimer's disease or somatoform pain disorder. In the following, a detailed summary of the contributions is given.

Preliminaries (Part I)

The preliminaries in Part I provides an introduction to the KDD process in general and discusses open challenges in clustering and biomedical imaging. In Section 2 we introduce common algorithms in clustering and classification as well as quality measures that are used in this thesis to quantify the quality of the results. Furthermore, we give an introduction to different imaging technologies that are used for clinical purposes.

Clustering Techniques (Part II)

Part II is dedicated to the development of new clustering methods. In particular, we focus on hierarchical and integrative clustering as well as clustering of complex objects like skyline object. The main advantages of the proposed methods can be summarized as follows:

- In the field of hierarchical clustering, ITCH (Information Theoretic Cluster Hierarchies) has been proposed in Section 3.2. This method overcomes several drawbacks of existing methods for hierarchical clustering. ITCH is built on a hierarchical variant of the information-theoretic principle of Minimum Description Length (hMDL). The hierarchical cluster structure is interpreted as a statistical model of the data and can thus be used for effective data compression by Huffman coding. The achievable compression rate induces the objective function for clustering. Using this objective function in combination with an hierarchical EM-like optimization technique, ITCH can find only only meaningful and valid clusters and handle outliers consistently by assigning them to an appropriate level of the hierarchy. ITCH is fully automatic and requires no difficult parameter setting. Furthermore, each cluster in the hierarchy is represented by an intuitive description. ITCH has been evaluated on several synthetic and real data sets, which shows that ITCH outperforms all competitors by magnitudes.

- In Section 3.3 we have proposed GACH (Genetic Algorithm for finding Cluster Hierarchies) which is based on a genetic algorithm (GA). A genetic algorithm is a stochastic optimization technique based on the mechanism of natural selection and genetics. A fitness function is used to select the best individuals in each generation. The hMDL criterion described in Section 3.2 is used as fitness function together with an GA based optimization technique which enables a more thorough exploration of the solution space to find the correct cluster structure. As GACH uses a MDL-based fitness function, it can be easily applied to real world applications without requiring any expertise about the data, like e.g. the real number of clusters. Due to the fact that GACH integrates an EM-like strategy, the content of all clusters is described by an intuitive description in form of a PDF. Outliers, i.e. data objects that to not belong to a certain cluster are assigned to appropriate inner nodes of the hierarchy, depending on their degree of outlierness. Our experimental evaluation demonstrates that GACH outperforms a multitude of other clustering approaches.

- The algorithm INTEGRATE has been introduced in Chapter 4 which is designed to cluster heterogeneous data that is described by numerical and categorical attributes. We have defined iMDL, an information theoretic clustering quality criterion suitable for integrative clustering. Using iMDL as objective function for clustering, INTEGRATE naturally balances the influence of numerical and categorical attributes without requiring any weighting parameters. We applied INTEGRATE to several data sets with mixed numerical and categorical attributes. The experiments show that INTEGRATE outperforms existing clustering methods for mixed type attributes.

- In Chapter 5, we use skylines as objects for data mining. Skylines give an optimized approximation of a data set w.r.t. two or more at-

tributes. Therefore, many techniques for determining the skyline of a data set have been proposed. We have defined a distance measure SkyDist that defines the distance between two skylines considering the dominance and non-dominance regions of both skylines. We propose two methods for computing SkyDist based on Monte-Carlo sampling (MCSkyDist) that gives an approximation of the distance and SPSkyDist that is based on the plane sweep paradigm and exactly determines the distance. In an extensive experimental evaluation, we demonstrate the efficiency of SkyDist and integrate SkyDist into different clustering algorithms to show the usefulness of SkyDist for a number of applications.

Techniques for Mining Biomedical Data (Part III)

In Part III we focus on the application of data mining techniques for biomedical applications. In particular, we apply techniques from feature selection, clustering, classification and graph mining to find patterns in the human brain that cause several neurological diseases.

- In Chapter 6, we have proposed the framework FCC to identify regions of high discriminatory power in high resolution magnetic resonance images of the brain. FCC combines data mining techniques from feature selection, clustering and classification. The results show excellent accuracies to identify Alzheimer's disease (AD) and Mild Cognitive Impairment (MCI) on high resolution MR images and indicate that it is a valuable complement to existing methods.

- In Chapter 7 we have presented the efficient toolkit JGrid, a distributed computing system that allows for a large scale comparative analysis on magnetic resonance images in order to get a deeper insight in diseases that cause abnormalities in the brain. JGrid provides a Java API that facilitates arbitrary software to be executed on a cluster of computers.

For this purpose, the FCC framework was implemented to conform to this API and applied to a set of 181 MR scans. The results show that an increase in efficiency of 80% could be achieved on a cluster of 10 computers and facilitates a not only effective but also efficient classification of several pathologies like Alzheimer's disease or schizophrenia.

- We applied several methods for motif discovery to task fMRI images of patients that suffer from somatoform pain disorder and healthy controls in Chapter 8. In particular, we applied the heuristic approach GREW and FANMOD, an exhaustive sampling method for frequent subgraph discovery to brain co-activation networks of 6 somatoform patients and 4 healthy controls. These motifs represent groups of brain compartments that covary in their activity during the process of pain stimulation. We found motifs that are characteristic for patients and healthy subjects, respectively.

9.2 Potentials for Future Work

At the end of this thesis, we describe some major directions for future research in the field of clustering and on mining biomedical data.

- ITCH and GACH, both rely on a coding scheme that is based on axis parallel Gaussian distribution functions. However, this is not the case in many real world applications. Extending the coding scheme to more general distribution functions would therefore be a next step.

- Ensembles are a simple but effective type of a multi-learner system, wherein each component tries to solve the same task. Cluster ensembles provide a tool for consolidation of results from a portfolio of individual clustering results. Many methods have recently been proposed for comparing different clustering results [158, 127, 49]. Combining several clusterings can lead to improved quality and robustness of the results

and has therefore high potential for future research. In this way, the results of ITCH and GACH can be combined to give more sophisticated results.

- INTEGRATE is based on a k-means algorithm. Finding the right k in partitioning clustering is an unsolved problem so far. INTEGRATE uses the iMDL criterion to select the correct k in a number of potential number of clusters. The baseline solution would be to try all k ranging from 1 to the number of points in the data set. However, this is not feasible for large data sets. One solution is, to try a sample of the data set but this is not satisfying either. Different strategies have been emerged in the last years for finding the right k [75, 33]. These strategies in combination with our new developed $iMDL$ criterion could give more precise results.

- Using the FCC framework on a set of MRI images has led to promising results for research of brain atrophy pattern in the brain of patients with Alzheimer's disease. To further validate the regions identified for AD and MCI we intend to apply or technique on a larger data set. Going beyond imaging, it is also very interesting to combine these findings with the clinical scores which are currently applied for diagnosis of AD. In addition, this or similar methods could successfully be applied to other pathologies, such es schizophrenia and other imaging modalities such es positron emission tomography.

- JGrid is designed for the analysis of MRI images. However, integrating other image modalities like e.g. positron emission tomography (PET) would have high potential for a better understanding of the anatomical changes in the human body. While imaging scans such as CT and MRI isolate organic anatomic changes in the body, PET is capable of detecting areas of molecular biology detail (even prior to anatomic change). PET scanning does this using radio-labeled molecular probes that have

different rates of uptake depending on the type and function of tissue involved. Changing of regional blood flow in various anatomic structures (as a measure of the injected positron emitter) can be visualized and relatively quantified with a PET scan. However, PET imaging is most useful in combination with anatomical imaging such as MRI, as in this way areas of abnormality on the PET images can be more perfectly correlated with anatomy on the MRI images.

- We applied several graph mining methods for frequent subgraph discovery to time-series data of patients with somatoform pain disorder and healthy controls. While these processes are dynamic, the models we considered were all static. In [171] a framework for finding dynamic motifs is proposed. Applying this framework to the task fMRI data will gain information about the temporal order of the motifs.

List of Figures

1.1 The KDD process. 4

2.1 Dendrogram for a sample data set. 21
2.2 Basic notations of DBSCAN. 22
2.3 Reachability plot of OPTICS for three Gaussian clusters. . . . 24
2.4 Slices of an MRI scan. 33

3.1 Optimization of the grid resolution for the hMDL criterion. . 49
3.2 Restructuring operations of ITCH. 58
3.3 Competitive evaluation of ITCH on DS_1. 60
3.4 Competitive evaluation of ITCH on DS_2. 62
3.5 Competitive evaluation of ITCH on glass data set. 64
3.6 Competitive evaluation of ITCH on breast cancer data set. . . 66
3.7 Stability of ITCH 68
3.8 `Mutation` operations of GACH 73
3.9 `Crossover` operator of GACH 74
3.10 Mean fitness of GACH w.r.t. the genetic parameters. 80
3.11 Competitive evaluation of GACH on DS_1 and DS_2. 83
3.12 Competitive evaluation of GACH on wine data set. 85

4.1 Running example for the algorithm INTEGRATE. 94
4.2 Evaluation of INTEGRATE on synthetic data. 103
4.3 Coding costs for different k values. 106

5.1 Motivating examples for SkyDist. 109
5.2 Dominance regions of two skylines X and Y. 114
5.3 Computation of SkyDist in 2-dimensional space. 116
5.4 Structure of a 3-dimensional skyline 117
5.5 Varying sample rate of MCSkyDist. 120
5.6 Clustering result on car data using SkyDist. 120
5.7 The dendrogram of Single Link using SkyDist in comparison to conventional metrics. 123
5.8 Clustering result on NBA data using SkyDist. 124

6.1 Features with high IG in MRI of AD vs. HC. 134
6.2 AD vs. HC: Discriminating regions in the brain. 138
6.3 MCI-AD vs. MCI-MCI: Discriminating regions in the brain. . 142
6.4 Effect of the parameter C on the classification accuracy of SVM. 144

7.1 Distributed architecture of JGrid 157
7.2 Graphical User Interface of JGrid 159
7.3 Status Dialog of JGrid. 160
7.4 Runtime analysis of JGrid. 163
7.5 I/O lock and waiting time analysis of JGrid. 163
7.6 Runtime of JGrid in different states of the workflow. 164

8.1 Brain network. 169
8.2 Transformation from a time series of three labeled graphs into the corresponding union graph. 174
8.3 Typical representatives of subgraphs found in brain network data. 180
8.4 Largest subgraphs found in brain network data. 181
8.5 Frequencies of ROIs in motifs of patients. 182
8.6 Frequencies of ROIs in motifs of controls. 182

List of Tables

3.1 Quantitative evaluation of ITCH on DS_1. 59
3.2 Quantitative evaluation of ITCH on DS_2. 61
3.3 Quantitative evaluation of ITCH on glass data set. 63
3.4 Quantitative evaluation of ITCH on cancer data set. 67
3.5 Quantitative evaluation of GACH on DS_1. 82
3.6 Quantitative evaluation of GACH on DS_2. 84
3.7 Quantitative evaluation of GACH on wine data set. 85

4.1 Results of INTEGRATE on real data. 105

5.1 Runtime analysis for SPSkyDist and MCSkyDist. 121

6.1 Demographic variables and MMSE for the different groups. . . 135
6.2 Different classification experiments on MRI data. 137
6.3 Classification results on MRI data. 139
6.4 Discriminative patterns in AD vs. HC. 140
6.5 Discriminative patterns in MCI-AD vs. MCI-MCI. 141

8.1 Statistics of the network models for each subject. 178
8.2 Motifs detected by FANMOD and GREW in fMRI data. . . . 184

Bibliography

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD Conference*, pages 207–216, 1993.

[2] A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63(2):503 – 527, 2007.

[3] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In *SIGMOD*, pages 49–60, 1999.

[5] L. G. Apostolova, C. A. Steiner, G. G. Akopyan, R. A. Dutton, K. M. Hayashi, A. W. Toga, J. L. Cummings, and P. M. Thompson. Three-dimensional gray matter atrophy mapping in mild cognitive impairment and mild Alzheimer disease. *Arch. Neurol.*, 64:1489–1495, Oct 2007.

[6] J. Ashburner, J. L. Andersson, and K. J. Friston. High-dimensional image registration using symmetric priors. *Neuroimage*, 9:619–628, Jun 1999.

[7] J. Ashburner and K. Friston. Multimodal image coregistration and partitioning–a unified framework. *Neuroimage*, 6:209–217, Oct 1997.

[8] J. Ashburner and K. J. Friston. Voxel-based morphometry–the methods. *Neuroimage*, 11:805–821, Jun 2000.

[9] A. Asuncion and D. Newman. UCI Machine Learning Repository, 2007.

[10] E. Bair, T. Hastie, D. Paul, and R. Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101:119–137, 2006.

[11] J. D. Banfield and A. E. Raftery. Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics*, 49(3):803–821, 1993.

[12] J. C. Baron, G. Chetelat, B. Desgranges, G. Perchey, B. Landeau, V. de la Sayette, and F. Eustache. In vivo mapping of gray matter loss with voxel-based morphometry in mild Alzheimer's disease. *Neuroimage*, 14:298–309, Aug 2001.

[13] S. Basu, M. Bilenko, and R. J. Mooney. A Probabilistic Framework for Semi-supervised Clustering. In *KDD*, pages 59–68, 2004.

[14] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979.

[15] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[16] M. Bilenko, S. Basu, and R. J. Mooney. Integrating Constraints and Metric Learning in Semi-supervised Clustering. In *ICML*, 2004.

[17] K. Blennow and H. Hampel. Csf markers for incipient alzheimer's disease. *The Lancet Neurology*, 2(10):605 – 613, 2003.

[18] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant. Robust Information-theoretic Clustering. In *KDD*, pages 65–75, 2006.

[19] C. Böhm, C. Faloutsos, and C. Plant. Outlier-robust clustering using independent components. In J. T.-L. Wang, editor, *SIGMOD Conference*, pages 185–198. ACM, 2008.

[20] C. Böhm, F. Fiedler, A. Oswald, C. Plant, B. Wackersreuther, and P. Wackersreuther. ITCH: Information-Theoretic Cluster Hierarchies. In *ECML/PKDD (1)*, pages 151–167, 2010.

[21] C. Böhm, S. Goebl, A. Oswald, C. Plant, M. Plavinski, and B. Wackersreuther. Integrative parameter-free clustering of data with mixed type attributes. In *PAKDD (1)*, pages 38–47, 2010.

[22] C. Böhm, A. Oswald, C. Plant, M. Plavinski, and B. Wackersreuther. Skydist: Data mining on skyline objects. In *PAKDD (1)*, pages 461–470, 2010.

[23] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[24] M. Bozzali, M. Filippi, G. Magnani, M. Cercignani, M. Franceschi, E. Schiatti, S. Castiglioni, R. Mossini, M. Falautano, G. Scotti, G. Comi, and A. Falini. The contribution of voxel-based morphometry in staging patients with mild cognitive impairment. *Neurology*, 67:453–460, Aug 2006.

[25] D. J. Brenner and E. J. Hall. Computed tomography–an increasing source of radiation exposure. *N. Engl. J. Med.*, 357:2277–2284, Nov 2007.

[26] R. K. Brouwer. Clustering feature vectors with mixed numerical and categorical attributes. *IJCIS*, 1-4:285–298, 2008.

[27] N. E. Carlson, M. M. Moore, A. Dame, D. Howieson, L. C. Silbert, J. F. Quinn, and J. A. Kaye. Trajectories of brain loss in aging and the development of cognitive impairment. *Neurology*, 70:828–833, Mar 2008.

[28] L. Y. Carreon, S. D. Glassman, J. D. Schwender, B. R. Subach, M. F. Gornet, and S. Ohno. Reliability and accuracy of fine-cut computed tomography scans to determine the status of anterior interbody fusions with metallic cages. *Spine J*, 8:998–1002, 2008.

[29] A. Chardin and P. Pérez. Unsupervised Image Classification with a Hierarchical EM Algorithm. In *ICCV*, pages 969–974, 1999.

[30] R. Chen and E. H. Herskovits. Network analysis of mild cognitive impairment. *Neuroimage*, 29:1252–1259, Feb 2006.

[31] G. Chetelat, B. Desgranges, V. De La Sayette, F. Viader, F. Eustache, and J. C. Baron. Mapping gray matter loss with voxel-based morphometry in mild cognitive impairment. *Neuroreport*, 13:1939–1943, Oct 2002.

[32] G. Chetelat, B. Landeau, F. Eustache, F. Mezenge, F. Viader, V. de la Sayette, B. Desgranges, and J. C. Baron. Using voxel-based morphometry to map the structural changes associated with rapid conversion in MCI: a longitudinal MRI study. *Neuroimage*, 27:934–946, Oct 2005.

[33] M. M.-T. Chiang and B. Mirkin. Intelligent choice of the number of clusters in -means clustering: An experimental study with different cluster spreads. *J. Classification*, 27(1):3–40, 2010.

[34] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting: Theory and optimizations. In *Intelligent Information Systems*, pages 595–604, 2005.

[35] R. Cilibrasi and P. M. B. Vitányi. Clustering by Compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[36] D. J. Cook and L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *J. Artif. Intell. Res. (JAIR)*, 1:231–255, 1994.

[37] M. L. Cooper. *Information measures for object recognition: accommodating signature variability.* PhD thesis, St. Louis, MO, USA, 1999. AAI9959927.

[38] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[39] J. G. Csernansky, L. Wang, J. Swank, J. P. Miller, M. Gado, D. McKeel, M. I. Miller, and J. C. Morris. Preclinical detection of Alzheimer's disease: hippocampal shape and volume predict dementia onset in the elderly. *Neuroimage*, 25:783–792, Apr 2005.

[40] C. Davatzikos, Y. Fan, X. Wu, D. Shen, and S. M. Resnick. Detection of prodromal Alzheimer's disease via pattern classification of magnetic resonance imaging. *Neurobiol. Aging*, 29:514–523, Apr 2008.

[41] C. Davatzikos, A. Genc, D. Xu, and S. M. Resnick. Voxel-based morphometry using the RAVENS maps: methods and validation using simulated longitudinal atrophy. *Neuroimage*, 14:1361–1369, Dec 2001.

[42] L. W. de Jong, K. van der Hiele, I. M. Veer, J. J. Houwing, R. G. Westendorp, E. L. Bollen, P. W. de Bruin, H. A. Middelkoop, M. A. van Buchem, and J. van der Grond. Strongly reduced volumes of putamen

and thalamus in Alzheimer's disease: an MRI study. *Brain*, 131:3277–3285, Dec 2008.

[43] C. DeCarli, D. G. Murphy, A. R. McIntosh, D. Teichberg, M. B. Schapiro, and B. Horwitz. Discriminant analysis of MRI measures as a method to determine the presence of dementia of the Alzheimer type. *Psychiatry Res*, 57:119–130, Jul 1995.

[44] D. Defays. An efficient algorithm for a complete link method. *Comput. J.*, 20(4):364–366, 1977.

[45] D. Dekker and P. H. J. Hendriks. Social network analysis. In *Encyclopedia of Knowledge Management*, pages 1460–1469. 2011.

[46] A. Demiriz, K. P. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. *Artificial neural networks in engineering*, pages 809–814, 1999.

[47] G. Demiröz and H. A. Güvenir. Classification by voting feature intervals. In *ECML*, pages 85–92, 1997.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. In *J Roy Stat Soc*, number 39, pages 1–31, 1977.

[49] B. Dom. An Information-Theoretic External Cluster-Validity Measure. In *UAI*, pages 137–145, 2002.

[50] B. Dubois, H. H. Feldman, C. Jacova, S. T. Dekosky, P. Barberger-Gateau, J. Cummings, A. Delacourte, D. Galasko, S. Gauthier, G. Jicha, K. Meguro, J. O'brien, F. Pasquier, P. Robert, M. Rossor, S. Salloway, Y. Stern, P. J. Visser, and P. Scheltens. Research criteria for the diagnosis of Alzheimer's disease: revising the NINCDS-ADRDA criteria. *Lancet Neurol*, 6:734–746, Aug 2007.

[51] S. Duchesne, C. Bocti, K. De Sousa, G. B. Frisoni, H. Chertkow, and D. L. Collins. Amnestic MCI future clinical status prediction using baseline MRI features. *Neurobiol. Aging*, 31:1606–1617, Sep 2010.

[52] S. Duchesne, A. Caroli, C. Geroldi, C. Barillot, G. B. Frisoni, and D. L. Collins. MRI-based automated computer classification of probable AD versus normal controls. *IEEE Trans Med Imaging*, 27:509–520, Apr 2008.

[53] S. Duchesne, A. Caroli, C. Geroldi, D. L. Collins, and G. B. Frisoni. Relating one-year cognitive change in mild cognitive impairment to baseline MRI features. *Neuroimage*, 47:1363–1370, Oct 2009.

[54] J. Duncan, R. J. Seitz, J. Kolodny, D. Bor, H. Herzog, A. Ahmed, F. N. Newell, and H. Emslie. A neural basis for general intelligence. *Science*, 289:457–460, Jul 2000.

[55] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, pages 226–231, 1996.

[56] M. Ewers, K. Buerger, S. J. Teipel, P. Scheltens, J. Schroder, R. P. Zinkowski, F. H. Bouwman, P. Schonknecht, N. S. Schoonenboom, N. Andreasen, A. Wallin, J. F. DeBernardis, D. J. Kerkman, B. Heindl, K. Blennow, and H. Hampel. Multicenter assessment of CSF-phosphorylated tau for the prediction of conversion of MCI. *Neurology*, 69:2205–2212, Dec 2007.

[57] Y. Fan, N. Batmanghelich, C. M. Clark, and C. Davatzikos. Spatial patterns of brain atrophy in MCI patients, identified via high-dimensional pattern classification, predict subsequent cognitive decline. *Neuroimage*, 39:1731–1743, Feb 2008.

[58] Y. Fan, D. Shen, R. C. Gur, R. E. Gur, and C. Davatzikos. COMPARE: classification of morphological patterns using adaptive regional elements. *IEEE Trans Med Imaging*, 26:93–105, Jan 2007.

[59] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[60] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, pages 82–88, 1996.

[61] L. Ferrarini, W. M. Palm, H. Olofsen, M. A. van Buchem, J. H. Reiber, and F. Admiraal-Behloul. Shape differences of the brain ventricles in Alzheimer's disease. *Neuroimage*, 32:1060–1069, Sep 2006.

[62] C. P. Ferri, M. Prince, C. Brayne, H. Brodaty, L. Fratiglioni, M. Ganguli, K. Hall, K. Hasegawa, H. Hendrie, and Y. Huang. Global prevalence of dementia: a Delphi consensus study. *The Lancet*, 366(9503):2112–2117, January 2006.

[63] M. F. Folstein, S. E. Folstein, and P. R. McHugh. "Mini-mental state". A practical method for grading the cognitive state of patients for the clinician. *J Psychiatr Res*, 12:189–198, Nov 1975.

[64] G. B. Frisoni, C. Testa, A. Zorzan, F. Sabattoli, A. Beltramello, H. Soininen, and M. P. Laakso. Detection of grey matter loss in mild Alzheimer's disease with voxel based morphometry. *J. Neurol. Neurosurg. Psychiatr.*, 73:657–664, Dec 2002.

[65] K. J. Friston, J. B. Poline, A. P. Holmes, C. D. Frith, and R. S. Frackowiak. A multivariate analysis of PET activation studies. *Hum Brain Mapp*, 4:140–151, 1996.

[66] C. Frost and C. Kallis. Reply: A plea for confidence intervals and consideration of generalizability in diagnostic studies. *Brain*, 132:e103; author reply e102, Apr 2009.

[67] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in Mathematical Sciences. W. H. Freeman, 1979.

[68] M. L. Gavrilova and J. G. Rokne. Reliable line segment intersection testing. *Computer-Aided Design*, 32(12):737–745, 2000.

[69] G. Gerig, O. Kubler, R. Kikinis, and F. A. Jolesz. Nonlinear anisotropic filtering of MRI data. *IEEE Trans Med Imaging*, 11:221–232, 1992.

[70] E. Geuze, H. Westenberg, A. Jochims, C. de Kloet, M. Bohus, E. Vermetten, and C. Schmahl. Altered Pain Processing in Veterans with Posttraumatic Stress Disorder. *Arch. Gen. Psychiatry*, 64:76–85, Jan 2007.

[71] J. Goldberger and S. T. Roweis. Hierarchical Clustering of a Mixture Model. In *NIPS*, 2004.

[72] C. D. Good, I. S. Johnsrude, J. Ashburner, R. N. Henson, K. J. Friston, and R. S. Frackowiak. A voxel-based morphometric study of ageing in 465 normal adult human brains. *Neuroimage*, 14:21–36, Jul 2001.

[73] P. Grünwald. A Tutorial Introduction to the Minimum Description Length Principle. *CoRR*, math.ST/0406077, 2004.

[74] H. Gündel, M. Valet, C. Sorg, D. Huber, C. Zimmer, T. Sprenger, and T. R. Tölle. Altered Cerebral Response to Noxious Heat Stimulation in Patients with Somatoform Pain Disorder. *Pain*, 137(2):413–421, July 2008.

[75] G. Hamerly and C. Elkan. Learning the k in k-means. In *NIPS*, 2003.

[76] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[77] O. Hansson, H. Zetterberg, P. Buchhave, E. Londos, K. Blennow, and L. Minthon. Association between CSF biomarkers and incipient Alzheimer's disease in patients with mild cognitive impairment: a follow-up study. *Lancet Neurol*, 5:228–234, Mar 2006.

[78] Z. He, X. Xu, and S. Deng. Clustering mixed numeric and categorical data: A cluster ensemble approach. *CoRR*, abs/cs/0509011, 2005.

[79] G. T. Herman. *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[80] S. K. Herukka, S. Helisalmi, M. Hallikainen, S. Tervo, H. Soininen, and T. Pirttila. CSF Abeta42, Tau and phosphorylated Tau, APOE epsilon4 allele and MCI type in progressive MCI. *Neurobiol. Aging*, 28:507–514, Apr 2007.

[81] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105, 1973.

[82] V. Hristidis, N. Koudas, and Y. Papakonstantinou. Prefer: A system for the efficient execution of multi-parametric ranked queries. In *SIGMOD Conference*, pages 259–270, 2001.

[83] C.-C. Hsu and Y.-C. Chen. Mining of mixed data with application to catalog marketing. *Expert Syst. Appl.*, 32(1):12–23, 2007.

[84] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov.*, 2(3):283–304, 1998.

[85] D. Ienco, R. G. Pensa, and R. Meo. Parameter-free hierarchical co-clustering by n-ary splits. In *ECML/PKDD (1)*, pages 580–595, 2009.

[86] A. Inokuchi, T. Washio, and H. Motoda. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. *Machine Learning*, 50(3):321–354, 2003.

[87] C. R. Jack, R. C. Petersen, Y. C. Xu, P. C. O'Brien, G. E. Smith, R. J. Ivnik, B. F. Boeve, S. C. Waring, E. G. Tangalos, and E. Kokmen. Prediction of AD with MRI-based hippocampal volume in mild cognitive impairment. *Neurology*, 52:1397–1403, Apr 1999.

[88] L. Jing, M. K. Ng, and J. Z. Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Trans. Knowl. Data Eng.*, 19(8):1026–1041, 2007.

[89] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *UAI*, pages 338–345, 1995.

[90] K. Kantarci and C. R. Jack. Neuroimaging in Alzheimer disease: an evidence-based review. *Neuroimaging Clin. N. Am.*, 13:197–209, May 2003.

[91] G. B. Karas, E. J. Burton, S. A. Rombouts, R. A. van Schijndel, J. T. O'Brien, P. Scheltens, I. G. McKeith, D. Williams, C. Ballard, and F. Barkhof. A comprehensive study of gray matter loss in patients with Alzheimer's disease using optimized voxel-based morphometry. *Neuroimage*, 18:895–907, Apr 2003.

[92] G. B. Karas, P. Scheltens, S. A. Rombouts, P. J. Visser, R. A. van Schijndel, N. C. Fox, and F. Barkhof. Global and local gray matter loss in mild cognitive impairment and Alzheimer's disease. *Neuroimage*, 23:708–716, Oct 2004.

[93] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient Sampling Algorithm for Estimating Subgraph Concentrations and Detecting Network Motifs. *Bioinformatics*, 20(11):1746–1758, Jul 2004.

[94] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[95] R. J. Killiany, B. T. Hyman, T. Gomez-Isla, M. B. Moss, R. Kikinis, F. Jolesz, R. Tanzi, K. Jones, and M. S. Albert. MRI measures of entorhinal cortex vs hippocampus in preclinical AD. *Neurology*, 58:1188–1196, Apr 2002.

[96] S. Klï¿½ppel, C. M. Stonnington, C. Chu, B. Draganski, R. I. Scahill, J. D. Rohrer, N. C. Fox, C. R. Jack, J. Ashburner, and R. S. Frackowiak. Automatic classification of MR scans in Alzheimer's disease. *Brain*, 131:681–689, Mar 2008.

[97] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, pages 275–286, 2002.

[98] H.-P. Kriegel, K. M. Borgwardt, P. Kröger, A. Pryakhin, M. Schubert, and A. Zimek. Future trends in data mining. *Data Min. Knowl. Discov.*, 15(1):87–97, 2007.

[99] K. Krishna and M. N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 29(3):433–439, 1999.

[100] M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. In *ICDM*, pages 313–320, 2001.

[101] M. Kuramochi and G. Karypis. Finding Frequent Patterns in a Large Sparse Graph. In *SDM*, 2004.

[102] M. Kuramochi and G. Karypis. GREW-A Scalable Frequent Subgraph Discovery Algorithm. In *ICDM*, pages 439–442, 2004.

[103] S. LaConte, S. Strother, V. Cherkassky, J. Anderson, and X. Hu. Support vector machines for temporal classification of block design fMRI data. *Neuroimage*, 26:317–329, Jun 2005.

[104] H. O. Lancaster. *The chi-squared distribution*. Wiley New York,, 1969.

[105] D. Le Bihan, J. F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabriat. Diffusion tensor imaging: concepts and applications. *J Magn Reson Imaging*, 13:534–546, Apr 2001.

[106] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636, 2006.

[107] T. Li and Y. Chen. A weight entropy k-means algorithm for clustering dataset with mixed numeric and categorical data. In *FSKD (1)*, pages 36–41, 2008.

[108] X. Lin, Y. Yuan, W. Wang, and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In *ICDE*, pages 502–513, 2005.

[109] L. A. N. Lorena and J. C. Furtado. Constructive Genetic Algorithm for Clustering Problems. *Evolutionary Computation*, 9(3):309–328, 2001.

[110] Z. Lu and T. K. Leen. Semi-supervised Learning with Penalized Probabilistic Clustering. In *NIPS*, 2004.

[111] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[112] U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.

[113] A. R. McIntosh, C. L. Grady, L. G. Ungerleider, J. V. Haxby, S. I. Rapoport, and B. Horwitz. Network analysis of cortical visual pathways mapped with PET. *J. Neurosci.*, 14:655–666, Feb 1994.

[114] M. J. McKeown, S. Makeig, G. G. Brown, T. P. Jung, S. S. Kindermann, A. J. Bell, and T. J. Sejnowski. Analysis of fMRI data by blind separation into independent spatial components. *Hum Brain Mapp*, 6:160–188, 1998.

[115] G. McKhann, D. Drachman, M. Folstein, R. Katzman, D. Price, and E. M. Stadlan. Clinical diagnosis of Alzheimer's disease: report of the NINCDS-ADRDA Work Group under the auspices of Department of Health and Human Services Task Force on Alzheimer's Disease. *Neurology*, 34:939–944, Jul 1984.

[116] K. Meguro, C. LeMestric, B. Landeau, B. Desgranges, F. Eustache, and J. C. Baron. Relations between hypometabolism in the posterior association neocortex and hippocampal atrophy in Alzheimer's disease: a PET/MRI correlative study. *J. Neurol. Neurosurg. Psychiatr.*, 71:315–321, Sep 2001.

[117] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer, 1996.

[118] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, Oct. 2002.

[119] C. Misra, Y. Fan, and C. Davatzikos. Baseline and longitudinal patterns of brain atrophy in MCI patients, and their use in prediction of short-term conversion to AD: results from ADNI. *Neuroimage*, 44:1415–1422, Feb 2009.

[120] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[121] T. Mori, M. Kikuchi, and K. Yoshida. Term weighting method based on information gain ratio for summarizing documents retrieved by ir systems. In *Journal of Natural Language Processing, 9(4):3–32*, 2003.

[122] J. C. Morris, A. Heyman, R. C. Mohs, J. P. Hughes, G. van Belle, G. Fillenbaum, E. D. Mellits, and C. Clark. The Consortium to Establish a Registry for Alzheimer's Disease (CERAD). Part I. Clinical and neuropsychological assessment of Alzheimer's disease. *Neurology*, 39:1159–1165, Sep 1989.

[123] J. Mourao-Miranda, A. L. Bokde, C. Born, H. Hampel, and M. Stetter. Classifying brain states and determining the discriminating activation patterns: Support Vector Machine on functional MRI data. *Neuroimage*, 28:980–995, Dec 2005.

[124] J. Mourao-Miranda, E. Reynaud, F. McGlone, G. Calvert, and M. Brammer. The impact of temporal compression and space selection on SVM analysis of single-subject and multi-subject fMRI data. *Neuroimage*, 33:1055–1065, Dec 2006.

[125] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *Comput. J.*, 26(4):354–359, 1983.

[126] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *VLDB*, pages 144–155, 1994.

[127] X. V. Nguyen, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML*, page 135, 2009.

[128] R. A. Novelline. *Squire's fundamentals of radiology*. Harvard University Press., 1997.

[129] T. O'Reilly. What is web 2.0, design patterns and business models for the next generation of software. Internet, September 2005.

[130] A. Oswald and B. Wackersreuther. Motif discovery in brain networks of patients with somatoform pain disorder. In *DEXA Workshops*, pages 328–332, 2009.

[131] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD Conference*, pages 467–478, 2003.

[132] S. C. Park, H. Shin, and B. K. Choi. A sweep line algorithm for polygonal chain intersection and its applications. In *IFIP WG5.2 GEO-6*, pages 187–195, 1998.

[133] D. Pelleg and A. W. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *ICML*, pages 727–734, 2000.

[134] C. Pennanen, M. Kivipelto, S. Tuomainen, P. Hartikainen, T. Hanninen, M. P. Laakso, M. Hallikainen, M. Vanhanen, A. Nissinen, E. L. Helkala, P. Vainio, R. Vanninen, K. Partanen, and H. Soininen. Hippocampus and entorhinal cortex in mild cognitive impairment and early AD. *Neurobiol. Aging*, 25:303–310, Mar 2004.

[135] C. Pennanen, C. Testa, M. P. Laakso, M. Hallikainen, E. L. Helkala, T. Hanninen, M. Kivipelto, M. Kononen, A. Nissinen, S. Tervo, M. Vanhanen, R. Vanninen, G. B. Frisoni, and H. Soininen. A voxel based morphometry study on mild cognitive impairment. *J. Neurol. Neurosurg. Psychiatr.*, 76:11–14, Jan 2005.

[136] F. Pernkopf and D. Bouchaffra. Genetic-Based EM Algorithm for Learning Gaussian Mixture Models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1344–1348, 2005.

[137] R. C. Petersen, R. Doody, A. Kurz, R. C. Mohs, J. C. Morris, P. V. Rabins, K. Ritchie, M. Rossor, L. Thal, and B. Winblad. Current concepts in mild cognitive impairment. *Arch. Neurol.*, 58:1985–1992, Dec 2001.

[138] B. Pham. Offset curves and surfaces: a brief survey. *Computer-Aided Design*, 24(4):223–229, 1992.

[139] G. Piatetsky-Shapiro, R. Grossman, C. Djeraba, R. Feldman, L. Getoor, and M. Zaki. Is there a grand challenge or x-prize for data mining? In *KDD*, pages 954–956, 2006.

[140] C. Plant, M. Osl, B. Tilg, and C. Baumgartner. Feature selection on high throughput seldi-tof mass-spectrometry data for identifying biomarker candidates in ovarian and prostate cancer. In *ICDM Workshops*, pages 174–179, 2006.

[141] C. Plant, S. J. Teipel, A. Oswald, C. Bohm, T. Meindl, J. Mourao-Miranda, A. W. Bokde, H. Hampel, and M. Ewers. Automated detection of brain atrophy patterns based on MRI for the prediction of Alzheimer's disease. *Neuroimage*, 50:162–174, Mar 2010.

[142] J. Platt. Machines using Sequential Minimal Optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[143] J. L. Price, P. B. Davis, J. C. Morris, and D. L. White. The distribution of tangles, plaques and related immunohistochemical markers in healthy aging and Alzheimer's disease. *Neurobiol. Aging*, 12:295–312, 1991.

[144] J. L. Price, A. I. Ko, M. J. Wade, S. K. Tsou, D. W. McKeel, and J. C. Morris. Neuron number in the entorhinal cortex and CA1 in preclinical Alzheimer disease. *Arch. Neurol.*, 58:1395–1402, Sep 2001.

[145] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[146] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[147] E. Rendon and J. S. Sánchez. Clustering based on compressed data for categorical and mixed attributes. In *SSPR/SPR*, pages 817–825, 2006.

[148] J. Rissanen. An introduction to the mdl principle. Technical report, Helsinkin Institute for Information Technology, 2005.

[149] Y. Sakaguchi, S. Ozawa, and M. Kotani. Feature extraction using supervised independent component analysis by maximizing class distance. volume 5, pages 2502–2506, 2002.

[150] G. E. Sarty. *Computing Brain Activity Maps from fMRI Time-Series Images*. Cambridge University Press, 2007.

[151] R. I. Scahill, J. M. Schott, J. M. Stevens, M. N. Rossor, and N. C. Fox. Mapping the evolution of regional atrophy in Alzheimer's disease: unbiased analysis of fluid-registered serial MRI. *Proc. Natl. Acad. Sci. U.S.A.*, 99:4703–4707, Apr 2002.

[152] P. Scheunders. A genetic c-Means clustering algorithm applied to color image quantization. *Pattern Recognition*, 30(6):859–866, 1997.

[153] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *Comput. J.*, 16(1):30–34, 1973.

[154] D. Skoutas, D. Sacharidis, A. Simitsis, and T. K. Sellis. Serving the sky: Discovering and selecting semantic web services through dynamic skyline queries. In *ICSC*, pages 222–229, 2008.

[155] N. Slonim and N. Tishby. Document Clustering using Word Clusters via the Information Bottleneck Method. In *SIGIR*, pages 208–215, 2000.

[156] S. Still and W. Bialek. How Many Clusters? An Information-Theoretic Perspective. *Neural Computation*, 16(12):2483–2506, 2004.

[157] T. R. Stoub, M. Bulgakova, S. Leurgans, D. A. Bennett, D. Fleischman, D. A. Turner, and L. deToledo Morrell. MRI predictors of risk of incident Alzheimer disease: a longitudinal study. *Neurology*, 64:1520–1524, May 2005.

[158] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[159] Y. Su, T. M. Murali, V. Pavlovic, M. Schaffer, and S. Kasif. RankGene: identification of diagnostic genes based on expression data. *Bioinformatics*, 19:1578–1579, Aug 2003.

[160] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.

[161] S. Teipel, M. Ewers, O. Dietrich, S. Schoenberg, F. Jessen, R. Heun, N. Freymann, H. J. Moller, and H. Hampel. [Reliability of multicenter magnetic resonance imaging. Results of a phantom test and in vivo measurements by the German Dementia Competence Network]. *Nervenarzt*, 77:1086–1092, Sep 2006.

[162] S. J. Teipel, C. Born, M. Ewers, A. L. Bokde, M. F. Reiser, H. J. Moller, and H. Hampel. Multivariate deformation-based analysis of brain atrophy to predict Alzheimer's disease in mild cognitive impairment. *Neuroimage*, 38:13–24, Oct 2007.

[163] S. J. Teipel, R. Stahl, O. Dietrich, S. O. Schoenberg, R. Perneczky, A. L. Bokde, M. F. Reiser, H. J. Moller, and H. Hampel. Multivariate network analysis of fiber tract integrity in Alzheimer's disease. *Neuroimage*, 34:985–995, Feb 2007.

[164] P. M. Thompson, K. M. Hayashi, G. de Zubicaray, A. L. Janke, S. E. Rose, J. Semple, D. Herman, M. S. Hong, S. S. Dittmer, D. M. Doddrell, and A. W. Toga. Dynamics of gray matter loss in Alzheimer's disease. *J. Neurosci.*, 23:994–1005, Feb 2003.

[165] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *CoRR*, physics/0004057, 2000.

[166] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated Anatomical Labeling of Activations in SPM using a Macroscopic Anatomical Parcellation of the MNI MRI Single-subject Brain. *NeuroImage*, 1(15):273–289, January 2002.

[167] N. Vasconcelos and A. Lippman. Learning Mixture Hierarchies. In *NIPS*, pages 606–612, 1998.

[168] P. Vemuri, J. L. Gunter, M. L. Senjem, J. L. Whitwell, K. Kantarci, D. S. Knopman, B. F. Boeve, R. C. Petersen, and C. R. Jack. Alzheimer's disease diagnosis in individual subjects using structural MR images: validation studies. *Neuroimage*, 39:1186–1197, Feb 2008.

[169] P. J. Visser, P. Scheltens, F. R. Verhey, B. Schmand, L. J. Launer, J. Jolles, and C. Jonker. Medial temporal lobe atrophy and memory dysfunction as predictors for dementia in subjects with mild cognitive impairment. *J. Neurol.*, 246:477–485, Jun 1999.

[170] E. M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Inf. Process. Manage.*, 22(6):465–476, 1986.

[171] B. Wackersreuther, P. Wackersreuther, A. Oswald, C. Böhm, and K. M. Borgwardt. Frequent subgraph discovery in dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 155–162, New York, NY, USA, 2010. ACM.

[172] S. Wernicke. Efficient Detection of Network Motifs. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(4):347–359, 2006.

[173] S. Wessely, C. Nimnuan, and M. Sharpe. Functional Somatic Syndromes: One or Many? *Lancet*, 354:936–939, September 1999.

[174] B. Winblad, K. Palmer, M. Kivipelto, V. Jelic, L. Fratiglioni, L. O. Wahlund, A. Nordberg, L. Backman, M. Albert, O. Almkvist, H. Arai, H. Basun, K. Blennow, M. de Leon, C. DeCarli, T. Erkinjuntti, E. Giacobini, C. Graff, J. Hardy, C. Jack, A. Jorm, K. Ritchie, C. van Duijn, P. Visser, and R. C. Petersen. Mild cognitive impairment–beyond controversies, towards a consensus: report of the International Working Group on Mild Cognitive Impairment. *J. Intern. Med.*, 256:240–246, Sep 2004.

[175] X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. In *ICDM*, pages 721–724, 2002.

[176] X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *KDD*, pages 286–295, 2003.

[177] Q. Yang and X. Wu. 10 challenging problems in data mining research. *IJITDM*, 5(4):597–604, 2006.

[178] J. Yin and Z. Tan. Clustering mixed type attributes in large dataset. In *ISPA*, pages 655–661, 2005.

[179] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD*, pages 103–114, 1996.

[180] Z. Zhong, M. Ewers, S. Teipel, K. Burger, A. Wallin, K. Blennow, P. He, C. McAllister, H. Hampel, and Y. Shen. Levels of beta-secretase (BACE1) in cerebrospinal fluid as a predictor of risk in mild cognitive impairment. *Arch. Gen. Psychiatry*, 64:718–726, Jun 2007.

[181] X. Zhu, R. Jin, Y. Breitbart, and G. Agrawal. Mmis07, 08: mining multiple information sources workshop report. *SIGKDD Explor. Newsl.*, 10(2):61–65, 2008.

[182] R. Zou and L. B. Holder. Frequent subgraph mining on a single large graph using sampling techniques. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 171–178, New York, NY, USA, 2010. ACM.