Multi-Purpose Exploratory Mining of Complex Data

Xiao He

München 2014

Multi-Purpose Exploratory Mining of Complex Data

Xiao He

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik

der Ludwig–Maximilians–Universität

München

vorgelegt von

Xiao He

aus Yangling, China

München, den 02 Sep 2014

Erstgutachter: Prof. Dr. Christian Böhm

Zweitgutachter: Prof. Xingquan Zhu

Tag der mündlichen Prüfung: 05 Nov 2014

Contents

Abstract **xvii**

1 Introduction **1**

1.1 Knowledge Discovery in Databases (KDD) 1

1.2 Exploratory Data Mining Tasks 3

1.2.1 Cluster Analysis 3

1.2.2 Frequent Pattern Mining 4

1.2.3 Dimensionality Reduction 4

1.2.4 Similarity and Dissimilarity Measure 4

1.3 Challenges in Explorative Mining Complex Data 5

1.3.1 High-dimensional Data 5

1.3.2 Complex Data 7

1.3.3 Parametrization 8

1.3.4 Interpretation 9

1.4 Contributions and Structure of the Thesis 9

2 Background **11**

2.1 Clustering 11

2.1.1 Flat Clustering 12

2.1.2 Hierarchical Clustering 14

2.2 Subspace Clustering 15

2.2.1 Axis-Parallel Subspace Clustering 16

2.2.2 Arbitrarily-oriented Subspace Clustering 20
2.2.3 Co-clustering 21
2.2.4 Biclustering 22
2.3 MDL-based Clustering 23
2.3.1 Minimum Description Length Principle 24
2.3.2 MDL-based Clustering 25
2.4 Evaluation of Clustering Results 26
2.4.1 Cluster Purity 26
2.4.2 Mutual Information 27
2.4.3 Precision, Recall and F-Measure 28

3 Relevant Overlapping Subspace Clusters on Categorical Data 31
3.1 Introduction 33
3.2 Optimization Goal Compression 34
3.2.1 Notations 36
3.2.2 Coding Scheme 36
3.3 Algorithm 39
3.3.1 Minimum Coding Problem 39
3.3.2 Algorithm ROCAT 41
3.4 Experiments 45
3.4.1 Synthetic Data 46
3.4.2 Real World Data 51
3.5 Related Work and Discussion 58
3.5.1 Categorical Subspace Clustering 58
3.5.2 Informative Pattern Mining 59
3.6 Conclusion 60

4 Multiple Subspace Selection for Hierarchical Clustering 61
4.1 Introduction 62

4.2 LDA and Orthogonal LDA . . . 66
4.3 Multiple Subspace Selection . . . 67
4.3.1 Orthogonal LDA-Kmeans . . . 68
4.3.2 Multiple Subspace Selection . . . 71
4.3.3 Runtime Complexity . . . 73
4.4 Experiments . . . 74
4.4.1 Setup. . . . 74
4.4.2 Comparing OLDA-Km with LDA-Km . . . 77
4.4.3 The effect of parameter Epsilon . . . 78
4.4.4 Clustering Quality . . . 79
4.4.5 A Case Study on Pendigits Data . . . 80
4.4.6 Scalability . . . 85
4.5 Related Work and Discussion . . . 86
4.5.1 Exploiting Supervised Techniques for Clustering . . . 86
4.5.2 Subspace Clustering . . . 88
4.5.3 Hierarchical Clustering . . . 88
4.6 Conclusion . . . 89

5 Summarization-Compression Miner 91

5.1 Introduction . . . 92
5.1.1 Contributions . . . 93
5.2 Compressing a Bipartite Graph . . . 94
5.2.1 Coding Scheme . . . 95
5.2.2 Hidden Relations Between Vertices . . . 97
5.3 Algorithm SCMiner . . . 100
5.4 Experiments . . . 103
5.4.1 Clustering Quality . . . 105
5.4.2 Hidden Structure . . . 109
5.4.3 Link Prediction Accuracy . . . 111

5.5 Related Work . . . 113
5.5.1 Co-clustering . . . 113
5.5.2 Graph Compression and Summarization . . . 114
5.5.3 Link Prediction . . . 115
5.6 Conclusion . . . 115

6 Probabilistic Integral Metric for Multi-instance Data 117
6.1 Introduction . . . 118
6.1.1 Motivation . . . 119
6.1.2 Goal . . . 121
6.1.3 Idea of our Technique PIM . . . 122
6.1.4 Contributions . . . 123
6.2 Probabilistic Integral Metric . . . 124
6.2.1 The Generative Model . . . 125
6.2.2 Our Similarity Metric . . . 126
6.2.3 Efficient Evaluation of PIM . . . 131
6.2.4 Monte Carlo Integration . . . 133
6.3 Complexity and Index Support . . . 134
6.4 Experiments . . . 135
6.4.1 Effectiveness of PIM . . . 135
6.4.2 Efficiency of Indexing with PIM . . . 143
6.5 Related Work and Discussion . . . 148
6.5.1 Similarities for Multi-instance Data . . . 149
6.5.2 Indexing Multi-instance Data . . . 150
6.5.3 Approaches for Uncertain Data . . . 151
6.5.4 Multi-instance and Metric Learning . . . 152
6.6 Conclusion . . . 152

7 Conclusion and Future Work **155**

7.1 Parameter-free Relevant Subspace Clustering 155

7.2 Hierarchical Visualization for Subspace Clusters 156

7.3 Summarization-based Co-clustering 157

7.4 Mining Multi-instance Data 158

Acknowledgments **173**

List of Figures

1.1 Knowledge Discovery in Databases (KDD) process. 2
1.2 Subspace clustering examples. 6
1.3 Complex high-dimensional data types. 7

3.1 Using compression to evaluate categorical clustering. 35
3.2 Searching phase of ROCAT. 43
3.3 4 processing candidates in Combining phase of ROCAT. 44
3.4 Synthetic categorical data for subspace clustering. 47
3.5 Robustness against outliers, $syn1$ to $syn4$ with outliers from left to right. . 50
3.6 Scalability of ROCAT and comparisons. 51

4.1 An example data set for illustrating the idea of hierarchical visualization. . 64
4.2 Hierarchical clustering and visualization. 65
4.3 The different objective function values on a 3D example data. 70
4.4 The hierarchical structure of dataset $Clus10$. 75
4.5 The effects of parameter ϵ on synthetic and real dataset for MSS. 78
4.6 The effects of irrelevant dimensions. 80
4.7 Hierarchy of Pendigits detected by MSS. 82
4.8 Hierarchical visualization on subspaces of Pendigits dataset found by MSS. 83
4.9 Reachability plot from HiCO on Pendigits with $\mu = 5$ and $k = 20$, NMI=0.271. 84
4.10 Contribution histograms of original features for subspaces in Figure 4.8. . . 85
4.11 The scalability of MSS and comparisons regarding Data size. 86

4.12 The scalability of MSS and comparisons regarding dimensionality. 87

5.1 Summarization and compression of a bipartite graph. 96

5.2 The strategy of SCMiner used for merging nodes 98

5.3 Results of SCMiner on synthetic data sets for various ϵ. 106

5.4 Cluster purity comparison on Movielens Data. 108

5.5 Hidden Structure Detected by SCMiner and CA. From Left to Right: Data Set BP1, BP2, BP3. 109

5.6 Hidden structure of cities detected by CA (left) and SCMiner (right). For SCMiner results, Square C represents a cluster consisting of capitals, Square F financial cities, Square E economic cities and Square W is the isolated city Washington, DC. On the other side, the cluster represented by Circle C mainly contains accountancy firms, Circle AB advertising and banking companies, Circle LB banking and law firms, and Circle L law firms. 111

5.7 Hidden structure of MovieLens detected by CA (left) and SCMiner (right). For SCMiner results, Square $O1$ and $O2$ denote clusters containing old users, Square YW represents a cluster of young women, and Square $YM1$ and $YM2$ young man. On the other side, the cluster represented by Circle FM represents high scored movies, Circle CD comedy and drama, Circle CR action, romance and comedy. Circles AA and AS represent adventure, action, thriller movies and action, sci-fi, thriller movies, respectively. 112

6.1 Three multi-instance objects $(\mathscr{X}, \mathscr{Y}, \mathscr{Z})$, derived from two templates $(\mathscr{T}_1, \mathscr{T}_2)$ through a generative model. 120

6.2 Intuition of Definition 8: The template instance $\mathbf{t}$ is varied over the whole data space $\mathbb{R}^{d=2}$. Shown are four examples $(\mathbf{t}_1, \ldots, \mathbf{t}_4)$ with their Gaussian probability density functions and with those circles colored in red having $c_{\mathscr{X}}(\mathbf{t}_i, r) \neq c_{\mathscr{Y}}(\mathbf{t}_i, r)$ ("differently covered"). 128

6.3 Overlay of the Voronoi diagrams of multi-instance object $\mathscr{X}$ and $\mathscr{Y}$. 133

6.4 A convex combination cell is cut into two pieces by the orthogonal line in the middle between the instances. 134
6.5 Effects of σ and S of PIM on synthetic data sets introduced in Section 6.4.1 with increasing R. 137
6.6 Comparison of nearest neighbor accuracy on synthetic data introduced in Section 6.4.1 with increasing R. 138
6.7 Comparing the diversity of NBA teams. First column: Performance of NBA teams provided by the number of games won (normalized). Remaining columns: Diversity in team composition as measured by the average distances among the players (normalized). PIM is the only similarity measure reflecting the ranking in team performance in team diversity as it can be expected from domain knowledge. 140
6.8 Nearest Neighbor Query Accuracy on Face data. 141
6.9 MDS on subset of Face (bpm, ch4f, and cheyer). Each point represents a face image of one of these three persons. Some interesting images are displayed. Red points: images of bpm, green: ch4f, blue: cheyer. 142
6.10 Number of samples vs Metric property. 145
6.11 Scalability experiments on synthetic data sets in Table 6.2. 146
6.12 Average query processing time on Forest Covertype data. 148
6.13 Average query processing time on HOUSE data. 148

List of Tables

3.1 Cluster Quality for Synthetic Data (F-Measure). 48
3.2 Subspace Quality for Synthetic Data (F-Measure). 49
3.3 Cluster Quality for Real Data (Precision). 52
3.4 Results on Congressional Votes. 53
3.5 Results on Mushroom. 54
3.6 Results on Splice. 56

4.1 Datasets description. 76
4.2 Parameterizations on real datasets. 76
4.3 NMI between OLDA-Km and LDA-Km. 77
4.4 F-Measure between OLDA-Km and LDA-Km. 77
4.5 Clustering quality comparison (NMI) 79
4.6 Clustering quality comparison (F-Measure) 79

5.1 Synthetic bipartite graphs 104
5.2 Clustering Performance on Synthetic data. 105
5.3 Results on World Cities Data. 107
5.4 Link Prediction Performances. 113

6.1 Top 5 similar players of M. Jordan in NBA data. 139
6.2 Parameters for Synthetic Data Sets 144

Abstract

Due to the increasing power of data acquisition and data storage technologies, a large amount of data sets with complex structure are collected in the era of data explosion. Instead of simple representations by low-dimensional numerical features, such data sources range from high-dimensional feature spaces to graph data describing relationships among objects. Many techniques exist in the literature for mining simple numerical data but only a few approaches touch the increasing challenge of mining complex data, such as high-dimensional vectors of non-numerical data type, time series data, graphs, and multi-instance data where each object is represented by a finite set of feature vectors. Besides, there are many important data mining tasks for high-dimensional data, such as clustering, outlier detection, dimensionality reduction, similarity search, classification, prediction and result interpretation. Many algorithms have been proposed to solve these tasks separately, although in some cases they are closely related. Detecting and exploiting the relationships among them is another important challenge. This thesis aims to solve these challenges in order to gain new knowledge from complex high-dimensional data.

We propose several new algorithms combining different data mining tasks to acquire novel knowledge from complex high-dimensional data: ROCAT (Relevant Overlapping Subspace Clusters on Categorical Data) automatically detects the most relevant overlapping subspace clusters on categorical data. It integrates clustering, feature selection and pattern mining without any input parameters in an information theoretic way. The next algorithm MSS (Multiple Subspace Selection) finds multiple low-dimensional subspaces for moderately high-dimensional data, each exhibiting an interesting cluster structure. For

better interpretation of the results, MSS visualizes the clusters in multiple low-dimensional subspaces in a hierarchical way. SCMiner (Summarization-Compression Miner) focuses on bipartite graph data, which integrates co-clustering, graph summarization, link prediction, and the discovery of the hidden structure of a bipartite graph data on the basis of data compression. Finally, we propose a novel similarity measure for multi-instance data. The Probabilistic Integral Metric (PIM) is based on a probabilistic generative model requiring few assumptions. Experiments demonstrate the effectiveness and efficiency of PIM for similarity search (multi-instance data indexing with M-tree), explorative data analysis and data mining (multi-instance classification).

To sum up, we propose algorithms combining different data mining tasks for complex data with various data types and data structures to discover the novel knowledge hidden behind the complex data.

Zusammenfassung

Aufgrund der gestiegenen Leistungsfähigkeit von Datenerfassungs- und Datenspeichertechnologien werden stark wachsende Mengen von komplex strukturierten Daten gesammelt. Statt einfacher Darstellungen, die auf niedrigdimensionalen numerischen Merkmalen basieren, erstrecken sich solche Datensätze von hochdimensionalen Merkmalsräumen bis hin zu Graphdaten, die Beziehungen zwischen Objekten beschreiben. In der Literatur wurden bereits zahlreiche Ansätze für das Data Mining von einfachen numerischen Daten beschrieben. Nur wenige Ansätze behandeln die größere Herausforderung, Data-Mining-Verfahren für komplex strukturierte Daten zu definieren, wie zum Beispiel hochdimensionalen Vektoren mit nicht-numerischen Datentypen, Zeitreihendaten, Graphdaten und Multiinstanzdaten, bei denen jedes Objekt durch eine endliche Menge von Merkmalsvektoren dargestellt wird.

Die Literatur unterscheidet außerdem zwischen vielen wichtigen Aufgabenstellungen beim Data Mining, wie zum Beispiel Clustering, Outlier Detection, Dimensionsreduktion, Ähnlichkeitssuche, Klassifikation, Vorhersage und Interpretation der Ergebnisse usw. Viele publizierte Algorithmen lösen diese Aufgaben lediglich separat, obwohl sie in einigen Fällen eng miteinander verwandt sind. Das Erkennen und Ausnutzen solcher Beziehungen zwischen diesen Aufgaben des Data Mining ist eine weitere wichtige Herausforderung. Die vorliegende Arbeit hat zum Ziel, diese beiden Integrations-Aufgaben zu bewältigen, um neuartige Erkenntnisse aus komplexen Datensätzen gewinnen zu können.

Wir stellen mehrere neue Algorithmen vor, die verschiedene Data-Mining-Aufgaben vereinen: ROCAT (Relevant Overlapping Subspace Clusters on Categorical Data) erkennt

automatisch die relevantesten sich überlappenden Subspace Cluster in kategorischen Daten. ROCAT integriert Clustering, Feature Selection und Pattern Mining auf informationstheoretische Weise und benötigt keine Eingabeparameter. Der Algorithmus MSS (Multiple Subspace Selection) findet mehrere niedrigdimensionale Unterräume für mittel- bis hochdimensionale Daten, die jeweils eine interessante Clusterstruktur aufweisen. MSS visualisiert die Cluster in mehreren niedrigdimensionalen Unterräumen in hierarchischer Anordnung, um die Ergebnisse besser zu interpretieren. Basierend auf der Datenkompression konzentriert sich der Algorithmus SCMiner (Summarization-Compression Miner) auf bipartite Graphen und integriert Co-Clustering, Graph Summarization, Link Prediction und das Erkennen von verborgenen Strukturen in bipartiten Graphen. Außerdem schlagen wir das neue Ähnlichkeitsmaß PIM (Probabilistic Integral Metric) für Multiinstanzdaten vor. PIM basiert auf einem probabilistischen generativen Modell. Experimente zeigen die Wirksamkeit und die Effizienz von PIM für die Ähnlichkeitssuche (Indizierung mittels eines M-Baums), explorative Datenanalyse und Data Mining (Multiinstanzklassifikation).

Alle in dieser Arbeit vorgestellten Algorithmen kombinieren verschiedene Aufgaben des Data Mining von komplexen Daten, die aus unterschiedlichen Datentypen und Datenstrukturen bestehen. Das gemeinsame Ziel ist es, aus den komplexen Daten neue Erkenntnisse zu gewinnen.

Chapter 1

Introduction

Nowadays, we are in the era of data explosion. Due to the increasing power of data acquisition and data storage technologies, huge amounts of data are collected in every aspect of our lives. Obviously, analyzing such data manually exceeds the human capabilities. Therefore, the incremental demand for automatically discovering the useful information and knowledge from the data is imminent as well. The concept of Knowledge Discovery in Databases (KDD) has thus been evolved as an interdisciplinary field that automatically mine the hidden rules and/or patterns behind the data. The discovered knowledge can be used for many applications ranging from science exploration to social activities analysis.

In this chapter, the general concept of Knowledge Discovery in Databases (KDD) is introduced in Section 1.1. Afterwards, the exploratory data mining methods that are studied in this thesis are elaborated in Section 1.2. Then the current challenges encountered in complex data are depicted in Section 1.3. Finally, Section 1.4 concludes with an outline and contributions of the thesis.

1.1 Knowledge Discovery in Databases (KDD)

> ***Knowledge Discovery in Databases (KDD)*** *is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable*

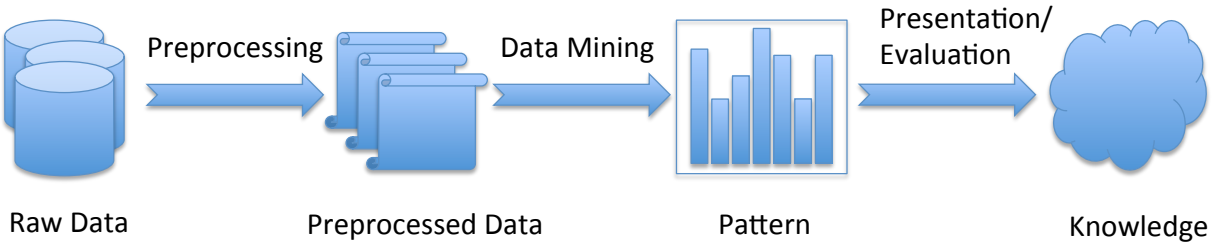


Figure 1.1: Knowledge Discovery in Databases (KDD) process.

patterns in data [44].

The KDD process is depicted in Figure 1.1 and described by a sequence of steps as follows:

1. **Preprocessing.** The raw data is preprocessed for mining, including data cleaning, integration, selection and transformation. In this step, firstly the raw data is cleaned by removing noises, handling missing entries, etc. Then data from multiple sources need to be integrated together. Afterwards subsets of data that are relevant to the mining task are selected. Finally, data is transformed to representations that are appropriate for mining by using feature selection or transformation methods. This step aims to increase the data quality to support the subsequent data mining step.

2. **Data mining.** The goal of this essential step is to extract unknown and useful patterns hidden behind the data using automatic algorithms. The algorithms that perform on the preprocessed data are chosen based on different data mining tasks, e.g. association rule mining, cluster analysis and classification.

3. **Presentation and evaluation.** The KDD process concludes with presenting the extracted patterns using knowledge visualization and representation techniques to the user. The user may return to previous steps if the result is not satisfactory.

The KDD process can involve iterations or loops between any two steps. Among them Data mining is the core step in KDD process, which is formally defined as follows.

> ***Data mining*** *is a step in the KDD process consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data [44].*

1.2 Exploratory Data Mining Tasks

In general, data mining tasks can be classified into two categories: descriptive and predictive [58]. Descriptive or exploratory mining tasks explore patterns from data and determine the properties, e.g. cluster analysis, association rule mining, outlier detection. Predictive mining tasks learn patterns from data and perform inference to make predictions, e.g. classification and prediction.

This thesis mainly focuses on descriptive/exploratory data mining tasks and their relationships, including cluster analysis, association rule mining, dimensional reduction and similarity measure. In the following, we introduce these tasks in more details.

1.2.1 Cluster Analysis

Clustering is the process of grouping the objects of a data set into clusters, so that the similarity between objects within a cluster is maximized and the similarity between objects in different clusters is minimized. Clustering is one of the main tasks for exploratory data mining, and it can be used to many research areas, including statistics, machine learning, information retrieval, pattern recognition and etc. Clustering is an unsupervised process that analyzes data objects without knowing class labels beforehand. Normally, the goal of clustering is to better understand and describe the data. Sometimes it is also the first step for other mining tasks such as generating class labels that facilitates the predictive mining tasks like classification.

1.2.2 Frequent Pattern Mining

Frequent pattern mining is the process of discovering patterns that frequently appear in a given data set. Frequent patterns express as itemsets, subsequences and substructures that occur in the same data satisfying a minimum support threshold. Finding such frequent patterns is a major factor in mining associations and correlations among data. Moreover, it helps other data mining tasks as well, such as rule-based classification and pattern-based clustering methods.

1.2.3 Dimensionality Reduction

Dimensionality reduction is the process of obtaining a reduced representation of the original data. It consists of feature selection and feature extraction. Feature selection methods aim at selecting subset of features so that irrelevant and redundant features are eliminated. Feature extraction methods transform the data from the original space to a lower dimensional space so that the most variances of data are preserved. Feature extraction methods can be both unsupervised (e.g. Principal Components Analysis) and supervised (e.g. Linear Discriminant Analysis). Traditionally, dimensionality reduction techniques are normally used in the data preprocessing step of the KDD process. However, sometimes global process cannot profit the following data mining tasks. Local dimensionality reduction is thus integrated into the data mining tasks, e.g. clustering, to obtain better results.

1.2.4 Similarity and Dissimilarity Measure

Similarity measure quantifies the similarity between two objects, which is usually an inverse of a distance or dissimilarity measure. It plays an extremely important role in exploratory data mining, e.g. clustering, since no labeling information is available. There are hundreds or even thousands of similarity measures for different types of data. For numerical data Minkowski distances are wide-spread and well-explored. However, mining complex

information-rich type data (e.g. mixed-type data, time-series data, multi-instance data) requires more studies on similarity measure.

1.3 Challenges in Explorative Mining Complex Data

The aim of this thesis is to propose novel algorithms that solve the challenges in clustering complex high-dimensional data by integrating different exploratory data mining methods introduced in Section 1.2. In the following, we will introduce some open challenges in clustering high-dimensional complex data. This thesis tries to address most of these challenges by proposing different data mining algorithms.

1.3.1 High-dimensional Data

Real-world data sets are normally represented as high-dimensional feature vectors. Traditional clustering approaches become difficult to handle such data sets due to the problem called "Curse of Dimensionality". With the increasing of dimensionality, similarity measures gradually loss their usefulness. The distances between objects are progressively getting closer and closer, thus all objects are nearly equidistant from each other and no cluster can be found in the full-dimensional space. Besides the extreme case of measuring similarity, "Curse of Dimensionality" affects many other aspects of clustering high-dimensional data, including irrelevant attributes, correlated attributes and overlapping clusters. For better illustrating these challenges, we use simple examples that is shown in Figure 1.2. Each of the three example data sets contains two attributes and each of them reflects the challenges mentioned above.

Irrelevant attributes. Objects in a cluster may only be homogeneous in a subset of attributes, while many other attributes are not relevant to the cluster. For example, in Figure 1.2(a) $C1$ is a full-dimensional cluster, while $C2$ is a subspace cluster that is only relevant to $Attribute1$. The existing of such so-called irrelevant attributes to clusters distorts the distances between objects in full-dimensional space. Besides, different clusters

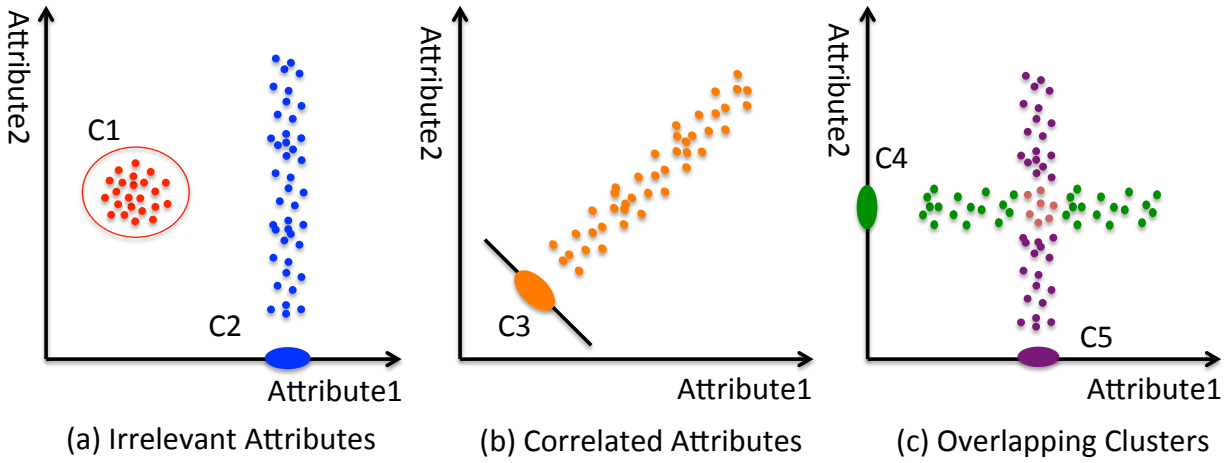


Figure 1.2: Subspace clustering examples.

may exist in different subsets of attributes or subspaces, thus some attributes may be irrelevant to one cluster but relevant to the other one. For instance, in Figure 1.2(a) *Attribute*2 is irrelevant to $C2$ but relevant to $C1$. Thus preprocessing data by globally feature selection technique is not useful in such cases.

Correlated attributes. Attributes that are relevant to a cluster may be correlated to or dependent on each other, which means that only selecting these attributes may not be helpful for detecting such clusters. Further rotation or transformation of the objects to an arbitrarily-oriented subspace is essential for clustering such data. Take Figure 1.2(b) for example, *Attribute*1 and *Attribute*2 are correlated and cluster $C3$ exists in an arbitrarily-oriented subspace. Same to irrelevant feature challenge, different correlations among the attributes may be relevant for different clusters as well. Therefore, global feature extraction technique (e.g. PCA) does not work in such cases.

Overlapping clusters. Different clusters may exist in different subspaces, thus there may be overlapping clusters. One data object can belong to one cluster in a certain subspace but to another cluster in a different subspace. In Figure 1.2(c), $C4$ and $C5$ overlap with each other, some objects belong to both clusters in different subspaces. Existing methods follow the idea of enumerating all the possible subspace clusters satisfying some input parameters. The number of such subspaces is exponential to the dimensionality.

Therefore, such approaches produce tremendous amount of redundant subspace clusters and highly depend on parameter settings.

1.3.2 Complex Data

Recently, a large amount of data sets have been generated in the field of biology, finance, market, multimedia and social network, etc. Most of these data sets are not only represented as high-dimensional numerical features, but complex and application specific data as well. Clustering such complex high-dimensional data meets further challenges.

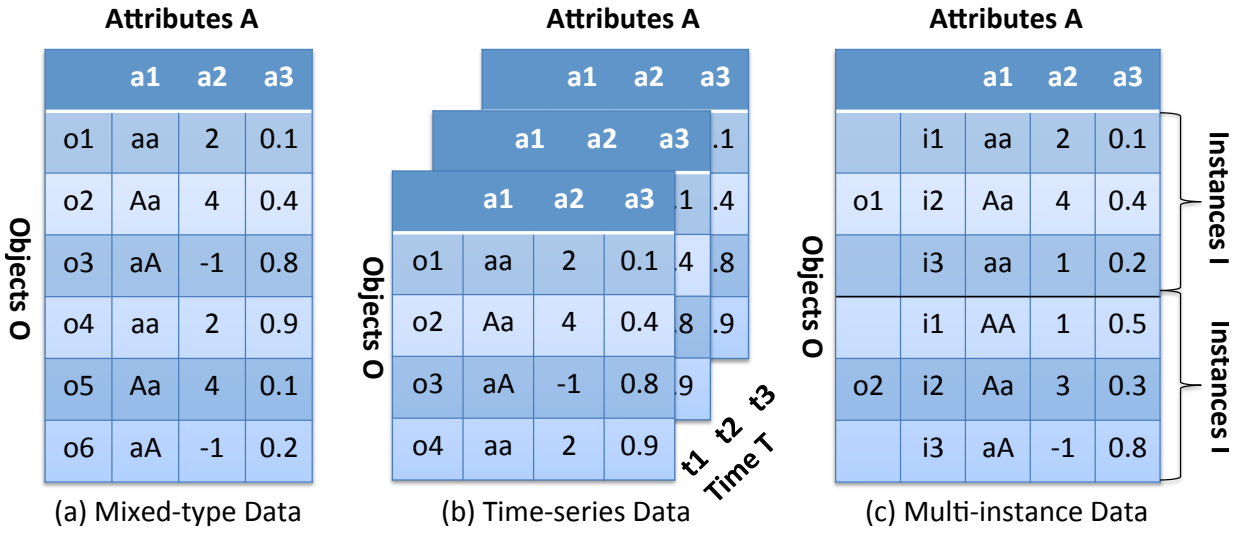


Figure 1.3: Complex high-dimensional data types.

Non-numerical data. Many data sets are collected in non-numerical forms, including categorical data, categorical/numerical mixed-type data (See Figure 1.3(a)), transactional data and relational data. Traditional numerical methods do not work on such data sets. In Categorical data the values of an attribute do not have any order and there is no distance information between them. e.g. attribute $a1$ in Figure 1.3(a). These two characteristics make it harder to cluster categorical data. Transactional data and relational data describe the relationship between two or more kinds of objects. If we treat the relationship as the attribute for one kind of objects, the dimensionality may be much higher than expected, which makes the subspace clustering problem hard to solve.

Multi-component data. Traditionally, we only process $Object \times Attribute$ data. However, in many applications data sets are represented in a higher order form. Multi-component data is $Object \times Attribute$ data extended with the third component. The entries of the third component are usually related to time or location, like time-series data ($Object \times Attribute \times Time$) with the third component related to time (See. Figure 1.3(b)) and multi-instance data ($Object \times Instance \times Attribute$) with the third component related to location (See Figure 1.3(c)). There are many challenges associated with clustering such Multi-component data.

Noisy and imperfect data. In real world data, there often exist some noisy objects or outliers due to the variate measurements or experimental errors. Besides, high-dimensional data may be contaminated with the noisy attributes as well. Many algorithms are very sensitive to noisy points or attributes. It is challenging to proposing subspace clustering algorithms that are robust to outliers and noisy attributes. In another side, imperfect data may contain missing or uncertain values due to the data collection condition which is ubiquitous in real world. Preprocessing data may easily remove the objects with missing values, however it might loss important information that owned by the objects for subspace clustering. Therefore, it is non-trial to handle imperfect data in subspace clustering as well.

1.3.3 Parametrization

Most clustering algorithms require the users to set parameters and output clusters satisfying the parameters. The performances of these algorithms strongly depend on suitable parameter settings. For instance, the number of clusters, the number of dimensionality of a subspace cluster or parameters that define a meaningful cluster, e.g. density, minimum number of objects and minimum distance threshold, need to be specified beforehand. These parameters are hard to estimate without a deep knowledge of the data sets. Some algorithms require many parameters, which make the tuning parameters process very complicated in real applications. Therefore, parameter-free algorithms or algorithms with robust parameters are essential in subspace cluster analysis.

1.3.4 Interpretation

High-dimensional data sets usually contain correlated and irrelevant features. In practice, meaningful clusters often exist in different arbitrarily-oriented subspaces of the original feature space. Analyzing and understanding the resulting clusters in arbitrarily-oriented subspaces is difficult. This is due to the fact that the clusters are usually not valid in the original feature space, and there is no semantic meaning for the resulting subspace. Further, each cluster may exist in a unique subspace, which makes it hard to analyze the relationship between clusters. Many techniques exist for detecting subspace clusters, however only very few approaches touch the challenging of interpreting them.

1.4 Contributions and Structure of the Thesis

The goals of this thesis are studying the relationship between exploratory data mining methods, integrating them into novel algorithms for clustering the aforementioned complex high-dimensional data, and performing extensive experiments to demonstrate the efficiency and effectiveness of the proposed algorithms. The following presents the major contributions and the general structure of the thesis.

Chapter 1 generally introduces the field of Knowledge Discovery in Databases and Data Mining to give basic context of the thesis. In addition, exploratory data mining tasks and new challenges in complex high-dimensional data that this thesis focuses are presented in this chapter.

Chapter 2 provides basic backgrounds of this thesis. The notions of clustering and subspace clustering in high-dimensional data are introduced. Besides, some state-of-art clustering and subspace clustering algorithms are surveyed in more details. Moreover, we provide some basic concept of Minimum Description Length principle, one of the most important techniques used in this thesis. Finally, the evaluation methods of clustering that are used in this thesis are presented.

Chapter 3 proposes ROCAT, a novel effective and efficient algorithm that detects the

most informative overlapping subspace clusters on categorical data. It integrates three exploratory data mining tasks: clustering, feature selection and pattern mining. By combining with the Minimum Description Length principle (MDL), it automatically determines a meaningful number of clusters to represent the data without any input parameter setting. ROCAT naturally avoids undesired redundancy of the result and guarantees that each detected cluster is relevant since it contributes to compress the data. Parts of the material presented in this chapter have been published in [60].

Chapter 4 presents a hierarchical subspace clustering algorithm MSS that finds multiple low-dimensional subspaces with correlated feature vectors, each exhibiting an interesting cluster structure. It includes hierarchical visualization with different low-dimensional subspaces for an intuitive interpretation of the clustering result. By doing this, MSS avoids of specifying the dimensionality of subspace, which is hard to estimate. Parts of the material presented in this chapter have been submitted in [61].

Chapter 5 introduces SCMiner, which focuses on relational bipartite graph data. It is a technique that integrates co-clustering, graph summarization, link prediction and the discovery of the hidden structure of a bipartite graph. It reduces a large bipartite input graph to a highly compact representation. In addition, while compressing the graph it detects the truly relevant clusters of both node types and their hidden relationships. Further, SCMiner does not rely on any input parameters that are difficult to estimate. Parts of material presented in this chapter have been published in [45].

Chapter 6 proposes a novel similarity measure PIM (Probabilistic Integral Metric) for multi-instance data. PIM is a highly effective and efficient metric that based on a probabilistic generative model requiring few assumptions held in many tasks: such as similarity search (multi-instance data indexing) and data mining (multi-instance classification). PIM scales linearly in the number of instances, thus is scalable to large data consisting of a high number of mutli-instance objects represented by massive amounts of instances. Parts of the material presented in this chapter have been submitted in [62].

Chapter 7 concludes the thesis and points out the possible future work.

Chapter 2

Background

The main goal of this thesis is to propose novel algorithms solving the clustering problem in complex high-dimensional data by combining different exploratory data mining techniques. This chapter gives a general background of the clustering problem. Section 2.1 starts with a general introduction to the clustering problem and briefly surveys the existing state-of-the-art algorithms. Then, Section 2.2 presents the subspace clustering problem, reviews various subspace clustering algorithms and analyzes their properties. After that, Section 2.3 introduces one of the most important techniques used in this thesis: Minimum Description Length (MDL) principle and reviews some MDL-based clustering algorithms. Finally, Section 2.4 introduces the common methods of evaluating the clustering results.

2.1 Clustering

Clustering is the process of grouping objects of a data set into clusters, so that the similarity between objects within a cluster is maximized and that between objects in different clusters is minimized.

A data set is normally represented as a matrix D with N rows and M columns. The set of rows in D corresponds to the objects set denoted by $O = \{O_1, O_2, ..., O_N\}$ and the set of columns corresponds to the attributes set denoted by $A = \{A_1, A_2, ..., A_M\}$.

The set A refers to a set of M-dimensional attributes and $V = \{V_1, V_2, ..., V_M\}$ denotes the corresponding domains. An object $o \in O$ is represented by a M-dimensional vector $v = \{v_1, v_2, ..., v_M\}$, where $v_i \in V_i$.

Given a data matrix D, a cluster C is defined as a subset of objects set O that exhibits similarly in attribute set A. To determine the similarity between objects in C, a similarity measure is essential.

In numerical data, Minkowski distance is wide-spread and well-explored. Given two M-dimensional objects $x = \{x_1, x_2, ..., x_M\}$ and $y = \{y_1, y_2, ..., y_M\}$ in D, the Minkowski distance is defined as follows:

$$dist(x, y) = (\sum_{i=1}^{M} |x_i - y_i|^p)^{1/p} \tag{2.1}$$

where $p > 0$ is the order of the distance. For $p > 1$ the Minkowski distance is a metric that satisfies the triangle inequality. Minkowski distance is typically used with p being one (Manhattan distance) or two (Euclidean distance).

For complex high-dimensional data, such as categorical data, time-series data or multi-instance data, Minkowski distance is not a good choice. Thus, it is non-trivial to define domain specific similarity measures for them. In Chapter 3 we introduce a novel method that uses coding length to measure the similarity of objects inside a cluster for categorical data. In Chapter 6 a new similarity measure for multi-instance data is introduced.

During the last decades, there are a large number clustering algorithms in the literatures, with multiple books, e.g. [67, 8], surveys, e.g. [20] and research papers, e.g. [99, 84, 100] to mention a few. In general, traditional clustering techniques can be divided into flat and hierarchical methods. In the following, we review the existing clustering approaches according to the category.

2.1.1 Flat Clustering

Many flat clustering algorithms exist in the literature. In this part, we briefly introduce the two most widely used type of flat clustering: partitioning clustering and density-based

clustering.

Partitioning Clustering

Partitioning clustering refers to the methods that divide data into K partitions, where each partition represents a cluster. Each object must be assigned to exactly one cluster and empty cluster is not allowed. Generally a global objective function is required to measure the quality of clustering, which usually refers to how similar or dissimilar are the objects within or between clusters. Even we know the number of clusters K before hand, achieving global optimum of partitioning clustering in numerical data is NP-hard [11]. Normally partitioning clustering techniques use the iterative relocation heuristic method [58] to achieve the local optimum. Such heuristic method first creates an initial partitioning, then tries to improve it by relocate objects from one group to another. K-means [83] is the most well-known and wide-spread partitioning clustering algorithm. K-means represents each cluster by the mean value of the objects inside the cluster, which is regarded as the cluster center. The sum of square errors for all objects to their cluster center are defined as the objective function. K-means algorithm performs by iteratively assigning all the objects to their nearest cluster center and updating the cluster center based on the assignment before. K-means is sensitive to noises and outliers, thus K-medoid [69] extends K-means against outliers, which represents each cluster by one of its objects that is closest to the cluster center. There are many other variations of K-means, e.g. K-modes [65] extends K-means for categorical data and EM [35] can be regarded as a soft version of K-means. EM models the input data as a mixture of k distributions and then iteratively improves the estimation of the distribution parameters to fit the data. The great benefits for K-means like algorithms are their efficiency. However, they need the number of clusters K to be specified in advance, which is usually hard in real applications.

Density-based clustering

Density-based clustering, e.g. DBSCAN [42], is another wide-spread type of flat clustering. In DBSCAN, clusters are defined as dense areas of objects which are separated by areas of lower object density. Clusters grow according to a density-based connectivity analysis, which is related to two input parameters ϵ and $MinPts$. An object is called a core object if there are at least $MinPts$ objects in its ϵ-neighborhood. Objects in the ϵ-neighborhood of a core object are directly density-reachable from the core object. DBSCAN searches for clusters by finding all the core object first, and then creates new clusters according to resulting core objects. After that it iteratively includes directly density-reachable objects from these core objects, which may merge connected clusters. The process terminates until no object can be added to any cluster. Usually, density-based clustering is used to find clusters with arbitrary shape and it is robust to noises and outliers. DBSCAN, DENCLUE [63] and OPTICS [14] are representative density-based clustering algorithms.

2.1.2 Hierarchical Clustering

Hierarchical clustering methods partition objects in different level and build a hierarchy of clusters. The resulting tree of clusters named dendrogram is commonly used to represent the process of hierarchical clustering. Strategies for hierarchical clustering generally belong to two categories: agglomerative and divisive. An agglomerative clustering works in a bottom-up merging way. Starting with singleton clusters (each cluster owns only one object), it recursively merge two or more most similar clusters until all objects belong to one cluster or a termination condition is satisfied. In comparison, a divisive clustering performs in a top-down splitting way. It starts with one cluster of all objects and recursively splits the clusters until all objects belongs to singleton clusters or a termination condition is satisfied. How to merge or split clusters is the most important component in hierarchical clustering, where a similarity measure between clusters is essential. Single Link [101] is a simple agglomerative hierarchical clustering algorithm that uses the minimum pair-wise distance between two clusters to measure their similarity. Variants of Single Link as Complete Link

and Average Link use different distance functions for pairs of clusters. Complete Link applies the maximum pair-wise distance between two clusters, while Average Link uses the average pair-wise distance between two clusters. Single-Link and Complete-Link take two extreme cases and Average Link is a compromise of them. One important issue of Single-Link and its variant hierarchical clustering methods is that once the clusters are merged or split it will process the newly generated cluster at the next step. If a bad merging or splitting decision is made, the error will accumulate and lead to a low-quality clustering. Besides, the resulting dendrogram always owns a large number of levels of clusters. How to choose the appropriate level is important in real application. Thus the termination condition is essential here as well.

2.2 Subspace Clustering

Traditional clustering algorithms are designed for clustering low-dimensional data and encounter many challenges facing high-dimensional data (See Section 1.3.1). In real world data, meaningful clusters often exist in different subspaces, due to the presence of irrelevant and correlated attributes. The subspace clustering algorithms are proposed to handle such challenges.

Given a data matrix D, a subspace cluster $SC = \{C, S\}$ is defined as a subset of objects set O that exhibits similarly in a subset of attributes set A, where $C \in O$ and $S \in T(A)$, where T is an orthogonal transformation.

During last decade, many subspace clustering algorithms are proposed to effectively and efficiently find such subspace clusters. Global dimensionality reduction techniques do not work in such case, since different clusters usually exist in different subspaces. Given a subspace of the original space, traditional clustering algorithms can be used to detect clusters. While given a cluster, dimensionality reduction techniques can be used to find the subspace of the cluster. Therefore, subspace clustering algorithms can be regarded as the combination of dimensionality reduction and clustering techniques that aim at detecting clusters and subspaces simultaneously.

Based on the properties of resulting subspaces where clusters exist, these algorithms can be categorized into: axis-parallel subspace clustering and arbitrarily-oriented subspace clustering [72]. Axis-parallel subspace clustering integrates with feature selection techniques and arbitrarily-oriented subspace clustering combines with feature transformation approaches. An axis-parallel subspace is composed of the subset of original attributes, while an arbitrarily-oriented subspace can be the subset of an arbitrary space that orthogonally transformed from the original attribute space.

Besides, there are two other branches of subspace clustering: Co-clustering and Biclustering. Although most of Co-clustering and Biclustering algorithms focus on axis parallel subspace clusters, they treat the subspace clustering problem in different approaches, thus we introduce them separately.

2.2.1 Axis-Parallel Subspace Clustering

Axis-parallel subspace clustering algorithms assume that clusters only exist in axis-parallel subspaces. Therefore, the number of possible axis-parallel subspaces is shrank down to finite compared with arbitrary subspaces. However, for a data set with M dimensions, there are $2^M - 1$ possible subspaces, which is exponential to the dimensionality. Thus, the naive way of performing clustering in all these subspaces is still intractable.

Based on the searching strategies, existing axis-parallel subspace clustering algorithms can be categorized into: dimension-growth subspace clustering and dimension-reduction projected clustering [58]. In the following, we will introduce them in more details.

Dimension-growth Subspace Clustering

Dimension-growth subspace clustering techniques use a bottom-up searching strategy. It usually starts at detecting single-dimensional subspace clusters and grows upward to higher-dimensional ones.

CLIQUE [10] is the first subspace clustering algorithm which solves the problem in a bottom-up way. Specifically, CLIQUE is a grid-based algorithm that partitions the data

space by a uniform grid into equally sized intervals of width ξ. If the number of data objects inside a unit of grid exceeds a density threshold τ, the unit is regarded as dense. CLIQUE takes the advantage of downward closure property of density, which means that if a k-dimensional cell is dense, all its projections in $(k-1)$-dimensional space are dense as well. Starting from detecting all the single-dimensional dense units, k-dimensional dense units are derived from the $(k-1)$-dimensional dense units until k equals to M. Then CLIQUE uses minimum description length principle [97] to further prune the dense units. Finally it identifies all the clusters in all axis-parallel subspaces. A Cluster is defined as the maximal set of connected dense units. ENCLUS [30] is a variant of CLIQUE, which are grid-based algorithm as well. Compared with CLIQUE, ENCLUS proposes three criteria based on the measure of entropy to identify subspaces with good clustering. Further, the downward closure property based on entropy and upward closure property based on dimensional correlation are used to prune the subspaces and enumerate all the clusters in the resulting axis-parallel subspaces. CLIQUE and ENCLUS are suffered from the fact that their performance in both effectiveness and efficiency are dependent on the granularity of the grid. A lower grid granularity performs fast but produces low quality clusters and vise verse.

MAFIA [89], another variant of CLIQUE, uses an adaptive grid to avoid the difficult input parameter of grid size. The adaptive grid size is determined based on the data distribution in a unique dimension. Specifically, MAFIA divides each dimension into many small intervals. Then it computes the histogram for each unit and merges adjacent units if they share similar histogram value. The process of generating subspace clusters is similar to CLIQUE. SUBCLU [73] uses a different way to avoid the grids, which uses DBSCAN [42] to create initial single-dimensional clusters. The resulting density-connected clusters satisfy the downward closure property as well. Thus SUBCLU further searches for subspace clusters in an Apriori style. The main drawback of SUBCLU is that it uses the same parameters ϵ and $MinPts$ for all the dimensions. It may result low quality initial clusters if the density of each dimension is diverse. And the error may accumulate in the following subspace clustering step.

Dimension-growth subspace clustering methods usually aim at enumerating all the clusters in all the possible axis-parallel subspaces. Since an object can be assigned to multiple subspace clusters, these techniques usually output huge number of subspace clusters. Many of these clusters are redundant and do not provide any further information. Besides, the huge number of resulting clusters makes it difficult for users to understand and interpret the results. Therefore, STATPC [87] and RESCU [88] are proposed to tackle this problem by mining the most interesting non-redundant subspace clusters. STATPC is based on the statistic and defines the significant region as the one that contains significantly more objects than others. RESCU defines an interestingness function for subspace clusters and a cost function of a clustering based on a coverage criterion. Further, the redundancy of a cluster given a clustering is defined as the cluster gain. Since deriving the global optimum result is NP-hard, RESCU greedily includes the cluster with most cluster gain. Both STATPC and RESCU reduce the size of output clusters and produce most interesting subspace clusters. However, their significant clustering models are dependent to many input parameters, which is hard to estimate for the user.

Dimension-reduction Projected Clustering

Dimension-reduction projected clustering aims at finding partitions of the dataset. Thus, overlapping clusters are not allowed for simplifying the problem. Each cluster is projected to a unique subspace with variable dimensionality. Normally, a dimension-reduction projected clustering method uses a specific distance function to determine the subspace of each cluster and integrates it into a full-dimensional clustering algorithm, e.g. K-means and DBSCAN.

PROCLUS [7] adopts the Manhattan Segmental distance and integrates it into K-medoid [69] clustering algorithm. Manhattan Segmental distance is the Manhattan distance on a subset of attributes. Firstly, PROCLUS determines a set of potential medoids and randomly chooses K from them as the medoids of each cluster. Then it assigns all the objects to their nearest medoids in the full-dimensional space. After that it iteratively up-

dates the subspace and medoid of each cluster and refines the clustering. For each cluster, a subset of attributes set is chosen as the subspace in the following way. If the distances of the objects along an attribute are smaller compared to statistical expectation, it belongs to the attributes subset. The old medoid is replaced with the new one from the set of potential medoids if the object function is decreased. Finally PROCLUS refines clustering by reassigning objects to subspace cluster using Manhattan Segmental distance. FINDIT [111] is a variation of PROCLUS, which uses additional heuristics to improve the efficiency and effectiveness. The biggest advantage of PROCLUS and FINDIT is their efficiency and scalability. However, it needs an input parameter which is the average number of attributes for subspaces, which is hard to estimate for the user.

PreDeCon [22] adopts a weighted Euclidean distance and integrates it into DBSCAN [42] clustering algorithm. The full-dimensional ϵ-neighborhood is used to determine the relevant attributes for each object. If the variance of distance along an attribute between an object and its neighbors is below a user-defined threshold δ, the attribute is considered relevant for the subspace preference of the object. The attributes relevant for the subspace preference of an object are weighted by a constant $\kappa \gg 1$ while the remaining attributes are weighted by 1. Then preference weighted ϵ-neighborhood, preference weighted reachability and subspace preference cluster are defined accordingly. Finally, DBSCAN is performed to find all the subspace clusters. Inherited from DBSCAN, PreDeCon does not need the number of clusters as the input parameter and handles outliers implicitly. However, PreDeCon requires the user to specify three input parameters that are usually hard to guess.

Dimension-reduction projected clustering approaches usually use a top-down strategy. PROCLUS first finds initial clusters in the full-dimensional space. Then it estimates the subspace of clusters and refines the clusters and subspaces in the following step. Similarly, PreDeCon uses full-dimensional ϵ-neighborhood to determine the relevant subspace for each object. When the dataset owns very high dimensionality, as mentioned before in Section 1.3.1, the global distances become closer and closer between all the objects. Therefore PROCLUS and PreDeCon loss their usefulness in very high-dimensional data or

dataset with too many irrelevant attributes. Furthermore, dimension-reduction projected clustering only partition the data set. Thus it cannot handle the case that overlapping clusters exist.

2.2.2 Arbitrarily-oriented Subspace Clustering

Arbitrarily-oriented subspace clustering methods assume that a cluster could exist in an arbitrarily-oriented subspace of the data space. The subspace can be regarded as the subset of an arbitrary space that is orthogonally transformed from the original attribute space. Thus, arbitrarily-oriented subspace clustering algorithms usually integrate the feature transformation techniques into traditional clustering algorithms.

ORCLUS [9] is the first arbitrarily-oriented subspace clustering approach which integrates Principal Component Analysis (PCA) into K-means. Given the number of clusters K, ORCLUS initially selects a large number of cluster centers. Data objects are assigned to these centers according to their distances in the corresponding transformed subspace (initially original space) to the center. The distance in the transformed subspace is computed via the eigenvectors inside clusters. Then the eigenvectors and clustering assignments are iteratively updated accordingly. After that, the number of clusters is reduced to K by iteratively merging nearest pairs of clusters and the dimensionality of each cluster is reduced to the user-specified dimensionality. ORCLUS needs user to specify the dimensionality of the resulting subspaces, which is hard to estimate. Moreover, all the clusters end with the same dimensionality of subspaces, which means that irrelevant transformed dimensions may be included. This phenomena affects the quality of clustering.

4C [23] is a density-based clustering paradigm that integrates PCA into DBSCAN. Specifically, the neighborhood of a point is obtained with the proposed distance in the eigensystems of two points. The eigensystem of an object $\vec{p}$ is based on the covariance matrix of the ϵ-neighborhood of $\vec{p}$ in original space. In the eigenvalue matrix $E_{\vec{p}}$, large eigenvalues are replaced by 1 and small eigenvalues are replaced by $\kappa \gg 1$. Then distance from an object $\vec{q}$ to $\vec{p}$ is calculated using the modified eigenvalue matrix $E'_{\vec{p}}$ of $\vec{p}$, which

is provided by $\sqrt{(\vec{p}-\vec{q})^T \cdot V_{\vec{p}} \cdot E'_{\vec{p}} \cdot V_{\vec{p}}^T \cdot (\vec{p}-\vec{q})}$. To make the distance symmetric, 4C calculate the distance from $\vec{p}$ to $\vec{q}$ analogously and picks the maximum of both distances. Finally, DBSCAN are performed to find all the arbitrarily-oriented subspace clusters. COPAC [4] is an extended work of 4C based on a similar idea. It avoids the problem of sparse ϵ-neighborhoods of points by taking a fixed number of k neighbors. Additional heuristics are used to choose a good k. Moreover, COPAC improves the efficiency and effectiveness of 4C.

The results of above arbitrarily-oriented subspace clustering algorithms are hard to interpret, since the detected subspaces are arbitrarily-oriented and there is no relationship among the clusters. LDA-Km [40] treated the problem in another way by integrating the supervised feature transformation technique Linear Discriminant Analysis (LDA) into K-means clustering. LDA-Km iteratively do LDA and K-means and finally the data are clustered while the subspace is selected simultaneously. LDA-Km detects a common subspace for all the arbitrarily-oriented subspace clusters, which is different from ORCLUS, 4C and COPAC, where each cluster owns a unique subspace. One great benefit of single subspace is the possibility to project all data to a joint space. If the joint space is low-dimensional, we can easily visualize the data by scatter plots. However, one drawback of LDA-Km is that LDA is not an orthogonal transformation like PCA, thus the resulting clusters are modified by scaling or warping. Besides, LDA-Km could not visualize the data in the projected space, if the dimensionality is higher than three.

2.2.3 Co-clustering

Co-clustering is a branch of subspace clustering which simultaneously partitions rows and columns of a data matrix.

Information-theoretic Co-clustering [38] (ITCC) is the pioneering co-clustering algorithm based on information theory. Specifically, ITCC simultaneously maps row elements to row clusters and column elements to column clusters, mutual information of each clustering state is calculated and compared to the initial state. Thus, the optimal co-clustering

result is obtained when the mutual information loss is minimal. ITCC iteratively adjust row clusters to maximize mutual information between row and column clusters followed by adjusting the column clusters in a similar way. The algorithm terminates until it reaches the local optimal where there is no improvement in mutual information.

Cross Association [29] is a parameter-free co-clustering method that processes a binary matrix and seeks clusters of rows and columns alternately. The matrix is divided into homogeneous rectangles that represent underlying structure of the data. It uses a top-down splitting fashion to achieve the optimal number of clusters. And clusters are refined by reassigning each individual rows and columns after splitting. The algorithm terminates when compression reaches a local minimum.

Long et al. [80] proposed a framework for co-clustering named block value decomposition (BVD), which formulates co-clustering as an optimization problem of matrix decomposition. It decomposes the data matrix into three components: row-coefficient, column-coefficient, and block value matrix. The final co-clusters is approached according to the decomposed matrices. In [79], Long et al. reconstructs a bipartite graph based on some hidden nodes and thus an optimal co-clustering result is obtained from the new bipartite graph which mostly approximates the original graph.

Compared with subspace clustering, co-clustering approaches usually only focus on the clustering part and ignore the relation between row and column clusters. The resulting clusters could not provide the information about in which subspace the clusters exist. This is one key point for subspace clustering.

2.2.4 Biclustering

Biclustering is another branch of subspace clustering that usually adopted in the analysis of gene expression data. In axis-parallel subspace clustering, subspace clusters are normally defined based on the density in subspaces. In compare, a bicluster is defined as the pattern in the gene expression data where a subset of genes share common behavior under a subset of conditions.

Cheng et al. [31] proposed the first biclustering algorithm for gene expression data. They defined the mean squared residue value of a row, a column and a bicluster to assess the quality of a bicluster. To find a bicluster, the whole data is regarded as the initial bicluster. Then they greedily remove the row or column with maximal mean squared residue value, until the value of bicluster satisfies a threshold δ. Finally, the procedure is iterated K times to discover K biclusters. The found clusters are masked with random values to enable the searching process in the next iteration. The main drawback of this algorithm is that the masking procedure may affect the results. Besides, it needs the number of clusters as the input parameter.

Wang et al. proposed pCluster model [109] that defines a bicluster as subset of genes exhibiting a coherent shifting pattern on a subset of conditions. The difference between two genes is measured by their relative differences of two conditions. Further, Wang et al. proposed a depth-first algorithm to mine pClusters. Specifically, they first find all the column-pair maximum dimension sets for all the gene-pairs. Then a pruning step is introduced to satisfying the user demands.

Ben-Dor et al. introduced a model, named OPSM (order preserving submatrix [17]), to discover a subset of genes ordered among a subset of the conditions. Thus the values in the selected conditions are strictly increasing in the bicluster. Further, the algorithm searches the best model in a greedy bottom-up approach. The best model is the one with largest statistical significance.

Compared with subspace clustering, biclustering usually defines different bicluster models under unique application.

2.3 MDL-based Clustering

As mentioned in Section 1.3, parametrization is one of the major challenges for clustering high-dimensional data. Many algorithms require some input parameters to be specified before hand, e.g. number of clusters, density threshold, distance threshold, .etc. To tackle such problem, information theoretic measures have been exploited as the clustering criteria

to choose the proper parameters. For instance, X-means [93] uses Bayesian Information Criterion (BIC) to choose the K in K-means. Besides BIC, there are numerous information theoretic criteria for model selection, such as Akaike Information Criterion (AIC) and Minimum Description Length principle (MDL). We use MDL in this thesis for choosing the best clustering model, because it is a general framework, and it does not specify any underlying truth assumption. In the following, we will firstly give a brief introduction of the concepts of MDL principle and then review some MDL-based clustering algorithms.

2.3.1 Minimum Description Length Principle

Minimum Description Length principle is a very important concept in information theory and a general method for inductive inference. The basic intuition behind it is described as follows.

> *MDL is based on the following insight: any regularity in the data can be used to compress the data, i.e. to describe it using symbols than the number of symbols needed to describe the data literally [54].*

Briefly, the better the data fits to the model, the more we are able to compress the data using the model. MDL connects learning the model from data to data compression. In the other way, we can say that the more we are able to compress the data, the more we have learned from the data.

For a given set of hypotheses H and data set D, MDL aims to find the hypotheses or combination of hypotheses in H that compresses D most. The earliest implementation of the idea is called two-part MDL, which is shown in Equation 2.2. We use two-part MDL, because it is simple and effective.

$$L(D, H) = L(H) + L(D|H) \tag{2.2}$$

Where $L(H)$ is the description length of the hypothesis and $L(D|H)$ is the description

length of data using the hypothesis.

MDL not only considers the compression of data with the model, but also the description of model itself. Therefore, it naturally avoids over-fitting problem with too complex model. Besides, it needs no underlying truth assumption, since it has a clear independent interpretation procedures. Therefore, MDL is perfect for choosing clustering model and solving the parameterization issue. The Chapter 3 and 5 we propose two MDL-based clustering algorithms for categorical data and bipartite graph data.

2.3.2 MDL-based Clustering

Recently, there are numerous MDL-based clustering algorithms, In the following, we will review a few of the typical ones.

Böhm et al. proposed RIC [21], a framework that can be used to purify and improve an initial clustering from any other algorithms. Based on MDL principle, they proposed Volume after Compression (VAC) criterion to measure the goodness of a clustering. They further proposed two novel algorithms: Robust fitting and Cluster merging to greedily reach a local minimum of VAC score. Therefore, the performance of RIC depends on the quality of the initial clustering, since it can only reach the local minimum.

Shao et al. proposed SynC and hSync [99] that exploit the synchronization phenomena for flat and hierarchical clustering. Specifically, they regard each object as a phase oscillator and simulate the dynamical behavior of the objects over time. MDl principle is further integrated to determine the best clustering model and meaningful hierarchical levels of abstraction.

Rakthanmanon et al. proposed a MDL-based clustering algorithm for time series data [95]. They showed that ignoring some data is critical for clustering time series streams. Further, MDL is proved to be very effective for comparing the stream data with different lengths or different sizes.

MDL has shown its effectiveness as a clustering criterion in many literatures. However, most subspace clustering algorithms still suffer from the parameterization issue. It is

definitely worth a try to integrate MDL principle into parameter-free subspace clustering.

2.4 Evaluation of Clustering Results

The goal of clustering is grouping similar objects into clusters. Therefore, a similarity measure between objects can be used as an internal criterion to measure the quality of clustering results. However, a clustering result with good internal criterion score may not indicate a good result in a specific application. In different applications, we must use different similarity measures as the internal criterion.

Another way to evaluate the clustering is using external criterion, which uses a set of gold standard classes labeled by domain experts. Then we evaluate the clustering by how well it matches the gold standard classes. There are many quantitative measures to compare the results of different clustering algorithms. In the following, three external evaluation measures (Cluster Purity, Mutual Information and Precision, Recall and F-Measure) that are used in this thesis are introduced.

2.4.1 Cluster Purity

Cluster Purity is the most straightforward external evaluation measure. Given a data set with N objects, a testing clustering result $T = \{T_1, T_2, ..., T_K\}$ and gold standard classes $G = \{G_1, G_2, ..., G_L\}$, each cluster is assigned to the class that dominate the cluster. Then the Cluster Purity is calculated by computing the percentage of correctly assigned objects as shown following:

$$Purity(T, G) = \frac{1}{N} \sum_{i=1}^{K} \max_{j \leq L} |T_i \cap G_j| \tag{2.3}$$

Purity ranges from 0 to 1, where 0 indicates worst clustering and 1 indicates perfect result. Achieving high purity is easy by producing a large number of clusters. In particular, accuracy is 1 when each object belongs to a unique cluster. Therefore, Cluster Purity is usually used to evaluate the clustering with fixed number of clusters.

2.4.2 Mutual Information

The mutual information between two random variables measures the dependence between them in an information theoretic way. The testing clustering result T and gold standard classes G can be regarded as two random variables, thus mutual information can be used to measure the information shared by them. The definition of mutual information is shown as below:

$$MI(T,G) = \sum_{i=1}^{K}\sum_{j=1}^{L} P(T_i \cap G_j) log \frac{P(T_i \cap G_j)}{P(T_i)P(G_j)} \tag{2.4}$$

where $P(T_i)$, $P(G_j)$ and $P(T_i \cap G_j)$ are the probabilities of an object being in cluster T_i, in class G_j and in the intersection of T_i and G_j, respectively.

Normalized Mutual Information (NMI) [105] is the most commonly used one, which is calculated as:

$$NMI(T,G) = \frac{MI(T,G)}{\sqrt{H(T)H(G)}} \tag{2.5}$$

where $H(T)$ and $H(G)$ are the entropies of clustering result T and gold standard G. $H(T)$ is defined as Equation 2.6 and $H(G)$ can be calculated similarly.

$$H(T) = -\sum_{i=1}^{K} P(T_i) log P(T_i) \tag{2.6}$$

The Normalized Mutual Information ranges from 0 to 1 as well, where 1 indicates that clustering result is identical to gold standard and 0 indicates that they are independent. NMI is normalized, thus it can be used to compare clusterings with different number of clusters.

Adjusted Mutual Information (AMI) [107] is another variant of mutual information measure. It is proposed to correct the measures for randomness. It is based on the expected mutual information and defined as follow:

$$AMI(T,G) = \frac{MI(T,G) - E(MI(T,G))}{\max(H(T), H(G) - E(MI(T,G)))} \tag{2.7}$$

where $E(MI(T,G))$ is the expected mutual information of clustering T and gold standard G. It is normalized as well and ranges from 0 to 1, where 1 indicates perfect result.

Adjusted Variation of Information (AVI) [107] is proposed to correct the randomness effect in clustering as well. It is based on the variation of information and defined as follow:

$$AVI(T,G) = \frac{2MI(T,G) - 2E(MI(T,G))}{H(T) + H(G) - 2E(MI(T,G))} \tag{2.8}$$

AVI takes value from 0 to 1 as well, where 1 indicates test clustering is identical to gold standard.

Mutual information based measurements treat the clustering result and gold standard as random variables and use their probability distribution to estimate the entropies and mutual information. Therefore, overlapping clusters are not allowed while using them, since the sum of probabilities of a random variable should be 1 ($\sum_{x \in X} P(x) = 1$). NMI, AMI and AVI are usually used to evaluate partitioning clusterings without overlapping clusters.

2.4.3 Precision, Recall and F-Measure

Precision, Recall and F-Measure are often used in the field of information retrieval for measuring binary classification. They can also be used to evaluate clustering by viewing it as a series of decisions, one for each of the $N(N-1)/2$ pairs of objects in the data [86]. It is also called pairwise Precision, Recall and F-Measure in some literature [15, 47].

A true positive decision (TP) assigns two objects to the same cluster when they belong to the same cluster in gold standard as well. A true negative decision (TN) assigns two objects to different clusters when they also belong to different clusters in gold standard. Besides, there are two types of errors. A false positive (FP) decision assigns two objects to

the same cluster when they belong to different clusters in gold standard. A false negative (FN) decision assigns two objects to different cluster when they belong to the same cluster according to gold standard. Then pairwise Precision and Recall are defined based on these decisions as follow:

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

F-Measure is defined as the harmonic mean of Precision and Recall:

$$FMeasure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.11}$$

F-Measure penalizes both false positive and false negative decisions during clustering. Besides, it can be used to evaluate clustering with overlapping clusters. When we process the pairwise decisions, we can handle the overlapping situation by such strategy. If two objects belong to at least one cluster in test clustering and gold standard, we treat them as true positive. Similarly, other kinds of decisions can be made.

Chapter 3

Relevant Overlapping Subspace Clusters on Categorical Data

As illustrated in Chapter 1, one major challenge for clustering high-dimensional data is the existence of irrelevant attributes. Existing dimension-growth methods, e.g. CLIQUE [10], SUBCLU [73], .etc, usually follow such idea. Firstly they define a subspace cluster with some parameters and then enumerate all the possible ones satisfying these parameters. However, in such way they produce a large number of clusters with high redundancy. Many of these clusters share the similar objects assignments and similar subspaces. They are all reported because they all fulfill the parameters. This makes it very difficult to interpret the results since there are too many clusters. We do not know which one is useful for our application and which one is more important. Further their performances strongly rely on parameters. Tuning parameters is not an easy work in real applications, which needs a large amount of manually work from domain experts.

Another challenge comes from the data type. In many applications, categorical data are collected, e.g. gene sequencing data. Due to missing order and spacing among the categories, selecting a suitable similarity measure is a difficult task. Many existing techniques require the user to specify input parameters that are difficult to estimate. Furthermore, compared to the large body of literature on clustering numerical data only relatively few

papers focus on subspace clustering of categorical data.

To tackle these challenges, in this chapter we propose ROCAT (**R**elevant **O**verlapping Subspace Clusters on **Cat**egorical Data), a novel technique based on the idea of data compression that detects most informative subspace clusters. Parts of the material presented in this chapter have been published in [60], where Xiao He was mostly responsible for the development of the main concept, implemented the main algorithms and wrote the largest parts of the paper; Claudia Plant supervised the project and made contributions to the building of coding scheme; Jing Feng and Son T.Mai helped with the implementation; Jing Feng and Bettina Konte performed part of experiments; The co-authors also contributed to the conceptual development and paper writing.

> *"Xiao He, Jing Feng, Bettina Konte, Son T.Mai and Claudia Plant. Relevant Overlapping Subspace Clusters on Categorical Data. The 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2014, in press."*

Following the Minimum Description Length principle, ROCAT automatically detects the most relevant subspace clusters without any input parameter. The relevance of each cluster is validated by its contribution to compress the data. Optimizing the trade-off between goodness-of-fit and model complexity, ROCAT automatically determines a meaningful number of clusters to represent the data. ROCAT is especially designed to detect subspace clusters on categorical data which may overlap in objects and/or attributes; i.e. objects can be assigned to different clusters in different subspaces and attributes may contribute to different subspaces containing clusters. ROCAT naturally avoids undesired redundancy in clusters and subspaces by allowing overlap only if it improves the compression rate. Extensive experiments demonstrate the effectiveness and efficiency of our approach.

The remainder of this chapter is organized as follows: In Section 3.1, it starts with an introduction. The optimization goal is elaborated in Section 3.2. Section 3.3 presents the algorithm ROCAT in detail. Section 3.4 contains an extensive experimental evaluation. Section 3.5 briefly discusses related work and Section 3.6 concludes the chapter.

3.1 Introduction

In many applications, ranging from social media to biomedicine, large categorical data sets are collected. The unique characteristic of categorical data is that the values of an attribute do not have any order. For example the attribute *genotype* having four values AA, Aa, aA and aa, where capital A represents the dominant normal variant of a gene and lowercase a the recessive version. There is no implicit order or quantitative spacing between the different categories.

To explore a large categorical data set, clustering is in principle very promising. Among the most successful approaches to unsupervised data mining, clustering aims at finding a natural partitioning of a data set into groups called clusters which represent the major patterns in the data. However, there are several special challenges associated with clustering moderate to high-dimensional categorical data:

1) Many existing algorithms require the user to choose a similarity metric and/or input parameters which are difficult to estimate, e.g. K-Modes [65], COOLCAT [16] and ROCK [55]. In comparison to numerical data where Minkowski distances are wide-spread and well-explored, the choice of a suitable similarity measure for categorical data is much more difficult: In a comparative survey [26], Boriah et al. studied the properties of 14 similarity measures and concluded that a suitable choice requires deep knowledge on the envisaged data mining task and the special characteristics of the data set to be analyzed. The same holds for input parameters like the number of clusters K in K-Modes [65], COOLCAT [16], SUBCAD [48], or the similarity threshold in ROCK [55].

2) Most techniques for clustering categorical data are limited to detect clusters in the full-dimensional space. For numerical data, the effects of the so-called *curse of dimensionality* have been extensively studied and many specialized techniques for clustering moderate to high-dimensional data have been proposed, for a survey see e.g. [72]. For categorical data, fewer approaches have been proposed, e.g. CACTUS [49], SUBCAD [48], and CLICKS [116], most of them associated with the problems mentioned above.

3) These methods are either partition-based (e.g. SUBCAD) or producing large re-

dundancies (e.g. CLICKS). STATPC [87] and RESCU [88] are proposed to find relevant non-redundant subspace clusters but are applicable to numerical data only. Detecting relevant overlapping subspace clusters on categorical data is an open research question.

To address these challenges, we propose a novel approach ROCAT (Relevant Overlapping Subspace Clusters on CATegorical data) combining the following benefits:

1. **Data compression as an intuitive notion of similarity.** Relating clustering to data compression, ROCAT considers the co-compressibility of the objects inside a cluster as one major aspect to evaluate the cluster quality. Thereby, ROCAT does not require the user to choose a similarity measure to quantify the pair-wise similarity among categorical data objects.

2. **Parameter-free detection of clusters.** Following the Minimum Description Length principle [97], co-compressibility is not the only aspect of cluster quality in ROCAT. We additionally consider the code length specifying the complexity of the clustering model and aim at minimizing both parts. Therefore, ROCAT is fully-automatic without requiring any parameters.

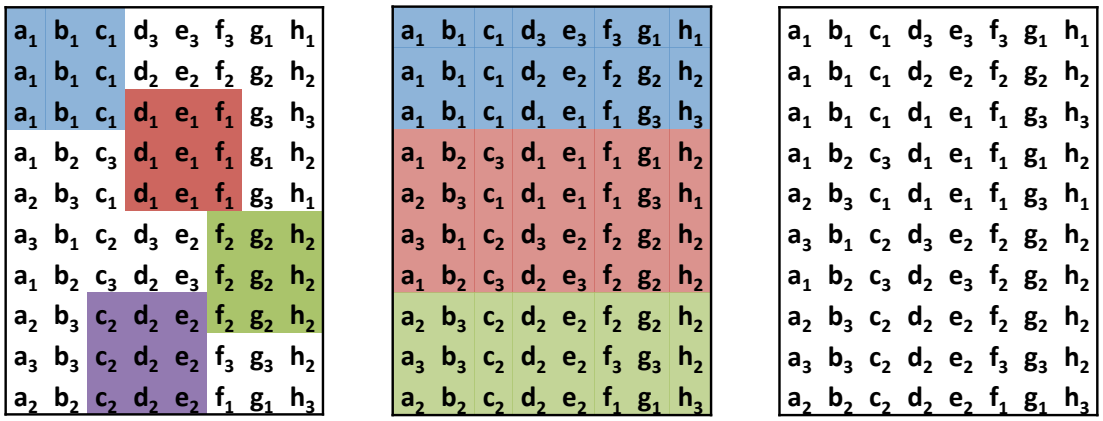
3. **Relevant overlapping subspace clusters.** The coding scheme of ROCAT allows overlapping in both objects and attributes set, but punishes redundancies. Therefore ROCAT finds the most relevant overlapping subspace clusters in the sense that they contribute to an effective compression of the data.

4. **Flexibly handling outliers.** ROCAT supports noise objects and noise attributes which are flexibly identified during the clustering process.

5. **Efficiency.** ROCAT scales linearly in data size.

3.2 Optimization Goal Compression

In this section, we elaborate how a subspace clustering can be used to effectively compress a categorical data set. The basic idea is that objects inside a cluster can be compactly



(a) Subspace Clustering (b) Full-D Clustering. (c) No Clustering.

Figure 3.1: Using compression to evaluate categorical clustering.

represented by joint coding in the corresponding subspace. Since subspace clusters may overlap, we validate the relevance of each cluster by its contribution to compress the data. Following the Minimum Description Length (MDL) principle [97], the clustering is regarded as a model for compression. The better the data fits to the model, the better is the compression rate since we only need to encode the deviations of the data from the model. In addition to the data, we also need to encode the model itself, which avoids overly complex models and naturally balances goodness-of-fit.

Figure 3.1 depicts an example of how we use compression to evaluate a clustering of categorical data. The data is represented by a matrix with 10 rows and 8 columns, each row represents 40 data objects and each column is an attribute. Figure 3.1(a) indicates a subspace clustering with 4 subspace clusters marked as colored squares. The clusters share the homogeneous data in the corresponding subspace, while the white area represents heterogeneous data, i.e. objects have arbitrary categories of the corresponding attributes. It costs 6076.5 bits to compress the data with the proposed coding scheme. Figure 3.1(b) and 3.1(c) depict a full-dimensional clustering and no clustering, where we need 6147.1 bits and 6670.9 bits to represent the whole data sets.

In the following, we firstly provide the necessary definitions and then propose a MDL-based coding scheme, which is specially designed for clustering categorical data.

3.2.1 Notations

Definition 1. *A **Categorical Data Set** is defined as $D = (X, V)$ with N objects and M attributes. $A_1, ..., A_M$ denote a set of categorical attributes and $V_1, ..., V_M$ a set of domains, where $V_j = \{V_{j_1}, ..., V_{j_m}\}$ is the domain for attribute A_j. $X \in N \times M$ is a matrix storing the categorical value $X(i, j)$ of object i in attribute A_j.*

Definition 2. *A **Subspace Cluster** $C_i = (X_i, V_i)$ is a subset of the data set $D = (X, V)$, where X_i is a sub-matrix of X and V_i is a subset of V.*

Definition 3. *A **Pure Subspace Cluster** is a Subspace Cluster where the objects share the same value in all its attributes.*

Definition 4. *A **Non-Clustered Area** S of data D modeled by subspace clusters $C_1, C_2, ..., C_K$ contains all the entries in matrix X that are not in any sub-matrix X_i, the white area in Figure 3.1(a).*

3.2.2 Coding Scheme

Following the concept of MDL principle, the quality of a model is provided by Eq. (3.1), where $L(H)$ denotes the cost for coding the model and $L(D|H)$ represents the cost of describing the data D under the model H.

$$L(D, H) = L(D|H) + L(H). \tag{3.1}$$

The model H contains K subspace clusters $C_1, C_2, ..., C_K$ and the non-clustered area S. According to our definition of a Subspace Cluster, $C_1, C_2, ..., C_K$ might overlap in both points and attributes set. The description of data D under the model H is provided by describing $C_1, C_2, ..., C_K$ and S separately. Therefore, $L(D|H)$ consists of two parts: the costs for the clusters and the non-clustered area.

$$L(D|H) = \sum_{i=1}^{K} CC_v(C_i) + CC_v(S). \tag{3.2}$$

In order to quantify the description length of a subspace cluster C_i or the non-clustered area S, we need to agree on an encoding scheme for C_i and S. For each cluster C_i, we encode the corresponding data sub-matrix X_i of C_i column by column. Specifically, for each assigned attribute A_j of cluster C_i, we calculate the probabilities for all categories in attribute A_j. Then any lossless coding method can be used to compress the column of attribute A_j in matrix X_i, i.e. Huffman coding. Practically we only need the coding length for evaluation, but not the true bits stream. Besides, lossless coding methods are lower bounded by the Shannon entropy. Therefore, we suggest to calculate the coding length of an attribute A_j in cluster C_i with categories $V_j = \{V_{j_1}, ..., V_{j_m}\}$ by the Shannon Entropy, which is defined as:

$$Entropy(P) = -\sum_{k=V_{j_1}}^{V_{jm}} \cdot P_k \cdot log_2 P_k, \tag{3.3}$$

$P = P_{V_{j_1}}, ...P_{V_{j_m}}$ are the probabilities of the categories in attribute A_j and cluster C_i, where $P_k = \frac{|X_i(:,j)==k|}{|C_i.obj|}$ and $|\cdot|$ is the number of entries in a set.

Then the coding cost for cluster C_i is provided as:

$$CC_v(C_i) = \sum_j |C_i.obj| \cdot Entropy(P). \tag{3.4}$$

The coding cost for the non-clustered area $CC_v(S)$ is calculated analogously to Eq. (3.4).

In addition to the data, we also need to describe the model itself $L(H)$. The model H contains the clustering assignments and probabilities that are used to encode C_i and S in Eq. (3.4). We need to describe both the object assignments CC_o and the attribute assignments CC_a to encode the clustering assignments for each subspace cluster C_i. In addition, we need to encode the probabilities used to describe data $L(D|H)$ since they are essential for lossless decoding. These probabilities are encoded as parameters CC_r.

$$L(H) = \sum_{i=1}^{K}(CC_o(C_i) + CC_a(C_i) + CC_r(C_i)) + CC_r(S). \tag{3.5}$$

Specifically, we describe the clustering assignments with object ID tables and attribute ID tables. The object ID table for a subspace cluster is a length N binary table. Objects are assigned 1 if they belong to the cluster, and 0 otherwise. As before, we suggest to encode the tables using an optimal Shannon code. The coding cost for an object ID table of C_i is calculated by its Shannon entropy:

$$CC_o(i) = -N \cdot (p(i) \cdot log_2 p(i) + n(i) \cdot log_2 n(i)), \tag{3.6}$$

where $p(i) = \frac{|C_i.obj|}{N}$ is the percentage of 1s, $n(i) = 1 - p(i)$ is the percentage of 0s. The coding cost for the attribute ID table is derived analogously to Eq. (3.6).

We encode the probabilities used in Eq. (3.4) as parameters. Following [97], the cost for the probabilities can be approximated by:

$$CC_r(i) = 0.5 \cdot |Param| \cdot log_2 |C_i.obj|, \tag{3.7}$$

where $|Param|$ is the number of parameters or probabilities. For each assigned attribute A_j of cluster C_i we need to encode $|V_j|$ probabilities. Therefore, $|Param| = \sum_j (|V_j|)$. The parameter cost for the non-clustered part $CC_r(S)$ is calculated analogously to Eq. (3.7).

The relevance of each cluster is validated by its contribution to compress the data. Thus the proposed coding scheme is perfectly suitable to evaluate relevant overlapping subspace clusters on categorical data. Firstly, it allows overlapping clusters in both objects and attributes set, but punishes those redundancies, since the overlapping parts will be encoded twice. Secondly, it avoids too complex models (too many small clusters) by encoding the model itself (clustering assignments and the probabilities), thus large informative clusters are preferred. In summary, clustering with the most relevant overlapping subspace clusters will achieve a lower coding cost under the proposed coding scheme.

3.3 Algorithm

In this section, we present an effective and efficient algorithm to identify the most relevant overlapping subspace clusters. Our optimization goal is to find the clustering model that best describes the categorical data set under the proposed coding scheme in Section 3.2.

3.3.1 Minimum Coding Problem

The proposed coding scheme does not specify how to find a good clustering; it can only say which of two clusterings is better. The problem, which we call the Minimum Coding Problem in the following, can be modeled as finding sub-matrices (Subspace Cluster) that allow the highest compression with respect to the proposed coding scheme. Given a data set D with N objects and M attributes, there are $I = (\sum_i^N \binom{N}{i}) \cdot (\sum_j^M \binom{M}{j})$ possible sub-matrices, further there are $\sum_i^I \binom{I}{i}$ possible clusterings with different combinations of sub-matrices. Obviously, an naive exhaustive search for the optimal result is infeasible even for a small data set, since the number of candidates $|I|$ is exponential to M and N. Even for the case that $|I|$ is polynomial to M and N, the Minimum Coding Problem is a NP-hard problem.

Minimum Coding Problem is NP-hard. The Set Covering Problem is known to be NP-hard [50]. Given a set of elements U and a set E of n sets, the Set Covering Problem finds smallest subsets of E whose union cover all the elements in U. The Minimum Coding Problem aims at finding sub-matrices of a data matrix X that cover all the entries of X, but uses a different kind of cost function, i.e. the proposed coding scheme. In the case that $|I|$ is polynomial and except of using a different cost function, the Minimum Coding Problem is equivalent to the Set Covering Problem or the Weighted Set Covering Problem. Since the proposed coding function can be calculated in polynomial time, the Minimum Coding Problem is NP-hard as well.

In summary, the Minimum Coding Problem is so difficult that we need an efficient and effective heuristic algorithm to achieve a local optimal result.

Algorithm 1 ROCAT

Input: Data set $D = (X, V)$
Output: Subspace clusters list $SubClus$

//Searching phase
$SubClus = \emptyset$; Queue $Matrix = \{D\}$;
while $Matrix \neq \emptyset$ **do**
  $Curr = Matrix$.Pop;
  $C =$ **FindBestPure**$(Curr, SubClus)$;
  if Eq. (3.1) decreases with $SubClus \cup C$ **then**
    $SubClus$.Add(C);
    $Matrix$.PushAll(**SplitSpace**$(Curr, C)$);
  end if
end while

//Combining phase
Priority queue $Pairs = \emptyset$;
for Each pair of clusters $C_i, C_j \in SubClus$ **do**
  $Overlap = |C_i.obj \cap C_j.obj| \cdot |C_i.att \cap C_j.att|$;
  If $Overlap > 0$, $Pairs$.Push(C_i, C_j);
end for
while $Pairs \neq \emptyset$ **do**
  Process $Pairs$.Pop as shown in Figure 3.3;
  Choose one process with minimum Eq. (3.1);
end while

//Reassigning phase
while Convergence **do**
  for Each cluster $C_i \in SubClus$ **do**
    Find all objects set O with same value in $C_i.att$;
    Assign or Remove O to $C_i.obj$ based on Eq.(3.1);
  end for
  for Each cluster $C_i \in SubClus$ **do**
    If $C_i.obj$ changed, Re-select $C_i.att$ based on Eq.(3.1);
  end for
end while

return $SubClus$

3.3.2 Algorithm ROCAT

The best-possible polynomial time approximation algorithm for the Set Cover Problem is the greedy algorithm [81]. At each stage, the set that contains the largest number of uncovered elements is selected. However, the greedy algorithm can not be used directly to solve the Minimum Coding Problem due to the following reasons. Firstly, the Minimum Coding Problem uses a different cost function, thus including the set that contains the largest number of uncovered elements may not reduce the proposed coding function. Secondly, the number of candidates $|I|$ is exponential to M and N, which makes the greedy algorithm exponential as well.

The proposed algorithm ROCAT is based on the greedy idea as well, but some essential modifications are made to solve the above two problems. Firstly, we need to iteratively include the Subspace Cluster that reduces the overall cost under the proposed coding function. Secondly, we need to reduce the number of candidate Subspace Clusters for greedy selection. Eq. (3.4) shows that including large Pure Subspace Clusters will lead to a reduction of the coding cost. Additionally, searching for large Pure Subspace Clusters will reduce the candidate space to polynomial as well. Therefore, we focus on selecting the Pure Subspace Clusters at first, then post-process them to get the final Subspace Clusters. The found Pure Subspace Clusters can overlap and exhibit redundancies. Therefore, during post-processing we firstly combine or split them to remove redundancies, then refine the results by locally modifying clustering assignments.

More precisely, there are three phases in ROCAT: Searching, Combining and Reassigning. Firstly, we iteratively include large Pure Subspace Clusters if the coding cost can be reduced in the Searching phase. Then we merge or split these candidates to remove redundancies in the Combining phase. The candidates with higher redundancy will be processed first. Finally, a reassignment step refines the result by reassigning objects and re-selecting the attributes to the candidate clusters. All phases are guided by the proposed coding scheme, and so every step guaranties decreasing coding cost, which finally leads to reaching a local minimum. The pseudocode of ROCAT is provided in Algorithm 1.

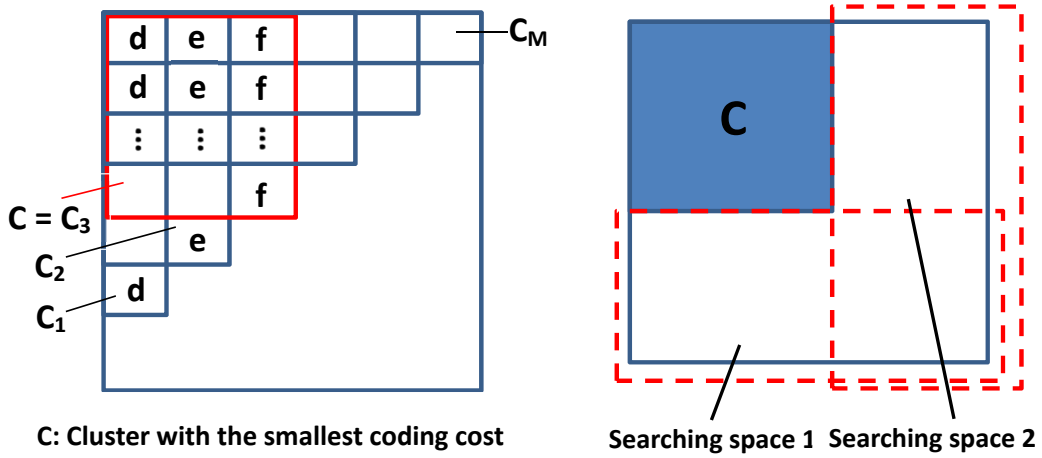
Algorithm 2 FindBestPure

Input: $Matrix$, $SubClus$
Output: C

$Att' = \emptyset$;
$PureClus = \emptyset$;
$Obj = Matrix.obj$;
$Att = Matrix.att$;
while $PureClus.Size < |Matrix.att|$ **do**
  Find $a \in Att$ with min Entropy regarding Obj;
  $Obj' = \{o \in Obj, X_{oa} = v, |o \in Obj| \text{ is max}\}$;
  $Att'.Add(a)$;
  Form cluster C with Obj' and Att';
  $PureClus.Add(C)$;
  $Att.Remove(a)$; $Obj = Obj'$;
end while
$C = \{C_i \in PureClus$, Eq.(3.1) is min for $(SubClus \cup C_i)\}$;

return C;

Searching Phase. We iteratively search for the best relevant Pure Subspace Cluster that reduces the coding cost most. The baseline coding cost is determined from a clustering model where all data points belong to the non-clustered area. Eq. (3.4) shows that large Pure Subspace Cluster will reduce the coding cost most, since the entropy of such clusters is 0. For a given searching matrix we find m large Pure Subspace Clusters, where m is the number of columns of the matrix. The pseudocode for this procedure called **FindBestPure** is depicted in Algorithm 2. The first cluster only contains attribute a with minimum entropy (see Eq. (3.3)) and objects with largest probability with respect to a. The second cluster is searched in the sub-matrix that contains the objects in the first cluster only. We expand the attribute set of the first cluster by the attribute that has the minimum entropy within the reduced data objects. This procedure is repeated until m Pure Subspace Clusters are found (see Figure 3.2a). Finally, the one that leads to minimum coding cost is returned as the best Pure Subspace Cluster. The first searching matrix is the value matrix X of data set D, in which we search for the best Pure Subspace



(a) Search the best pure cluster C.

(b) Split the search Space by C.

Figure 3.2: Searching phase of ROCAT.

Cluster C. If including C decreases the coding cost, we split the current searching matrix by C into two new ones (see Figure 3.2b) and add both to the searching matrix queue. We continue to search for best Pure Subspace Clusters until the searching queue is empty.

Combining phase. The Pure Subspace Clusters found in the Searching phase can overlap. We remove the redundancies in the Combining phase. The redundancy of each pair of clusters is modeled by their mutual information, which can be approximated by the overlapping entries between them. We firstly choose the two clusters C_i and C_j with the largest redundancy. Then we calculate the value of Eq. (3.1) for 4 different processing steps that are illustrated in Figure 3.3. We can preserve both clusters, combine the two clusters into one, preserve C_j and split C_i or preserve C_i and split C_j. Finally, we choose the step that yields the minimum coding cost. The phase is terminated when every pair of overlapped clusters has been processed.

Reassigning phase. The described Combining phase removes the redundancies by combining or splitting pairs of clusters only, thus redundancies may still exist among clusters. Besides, some objects may not be assigned to any cluster yet. Therefore we post-process the clusters in this phase to refine the result. Firstly, for each cluster C_i we fix the attributes set and adjust the objects set. In detail, objects set $O \subset D.obj$ with identical value in $C_i.att$ is assigned to C_i or removed from it if this decreases the coding cost.

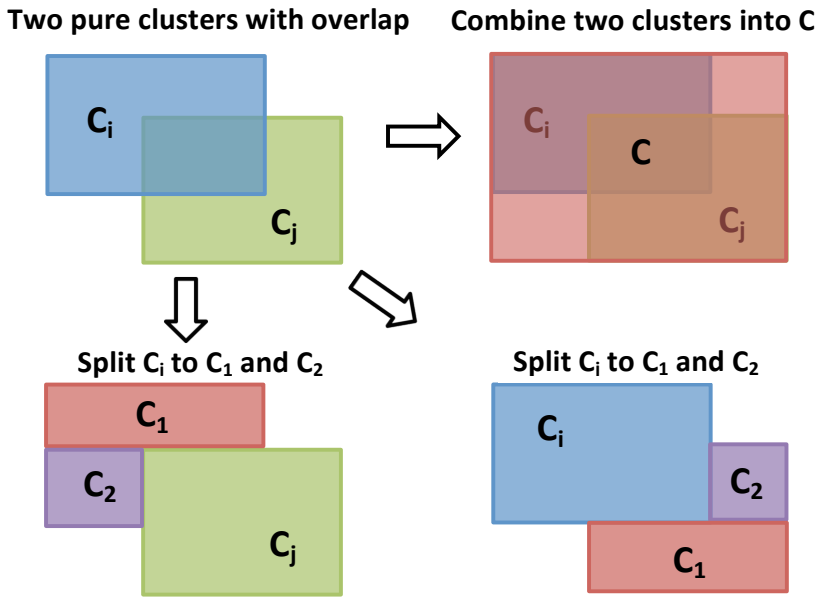


Figure 3.3: 4 processing candidates in Combining phase of ROCAT.

Secondly, we fix the objects set and try to improve the subspace of each cluster. Intuitively, data objects should be compact in the subspace of C_i and sparse in the remaining attributes. Therefore, we rank the attributes according to their entropy (see Eq. (3.3)), since a compact attributes set leads to a lower entropy while a sparse attributes set causes a higher entropy. Finally, we compare the coding cost for the top-ranked attributes sets and keep the best one if it yields an improvement over the old attributes set. We iteratively do the two steps until no attributes or objects set changes. Those objects that still can not be assigned to any cluster are naturally regarded as outliers.

Runtime Complexity. The runtime complexity of ROCAT for a data set with N objects and M attributes can also be divided into 3 parts. In the Searching phase, we need to go through β objects α times for each dimension, where $\beta \leq N$ and $\alpha \leq M$. Suppose we find γ clusters, controlled by MDL normally $\gamma \ll N, M$. Therefore the runtime complexity in this phase is $O(\alpha \cdot \beta \cdot \gamma \cdot M)$, which is equal to $O(M^2 \cdot N)$. In the Combining phase we need to go through all pairs of clusters, so the runtime complexity in this phase is $\mathrm{O}(\gamma^2 \cdot \kappa \cdot \iota)$, where $\kappa < N$ and $\iota < M$ are the average number of objects and attributes in each pure cluster. The runtime complexity in the Combining phase is equal to $O(M \cdot N)$. In the

Reassigning phase, in each iteration for each cluster we need to go through its objects set and attributes set once. Therefore the runtime complexity in this phase is $O(i \cdot (N \cdot M))$, which is equal to $O(M \cdot N)$ since the Reassigning phase normally converges very fast. The overall runtime complexity of ROCAT therefore is $O(M^2 \cdot N)$.

3.4 Experiments

In this section, we compare the performance of ROCAT to 8 methods from different areas which are related to this work. Firstly, we compare ROCAT to SUBCAD [48], CLICKS [116] and CLIQUE [10], 3 algorithms for subspace clustering on high-dimensional categorical data. CLIQUE is designed for numerical data but can be easily extended for categorical data. Moreover, we compare our work to two parameter-free algorithms for categorical data, DHCC [114] and AT-DC [28]. Due to space limitations, we do not compare to classical categorical clustering methods, i.e. K-modes [65], ROCK [55], COOLCAT [16]. These methods are not designed to find subspace clusters anyway and in addition DHCC [114] and AT-DC [28] have shown to yield better clustering models. Finally, we compare to 3 algorithms for informative itemset mining, Tiling [52], MTV [85] and Hyper+ [113], which try to find the most important itemsets. The itemset mining methods can be treated as categorical subspace clustering, since the detected itemsets can be regarded as the attributes sets of subspace clusters, while the objects that support the itemset forms the corresponding clusters. CLICKS and CLIQUE are based on the idea of itemset mining as well.

We implement ROCAT and SUBCAD in Java and use CLIQUE from the ELKI package [6]. The codes for all the other methods are provided by the authors. ROCAT, DHCC, AT-DC are parameter-free methods. SUBCAD and Tiling need the number of clusters K, where we set the true number for synthetic data and try different K for real data and output the best results. Besides, the performance of SUBCAD depends on its initialization, thus we report the average results of 10 runs. MTV is proposed as a parameter-free method, but as the execution time is too long and it allows the user to set the number of desired

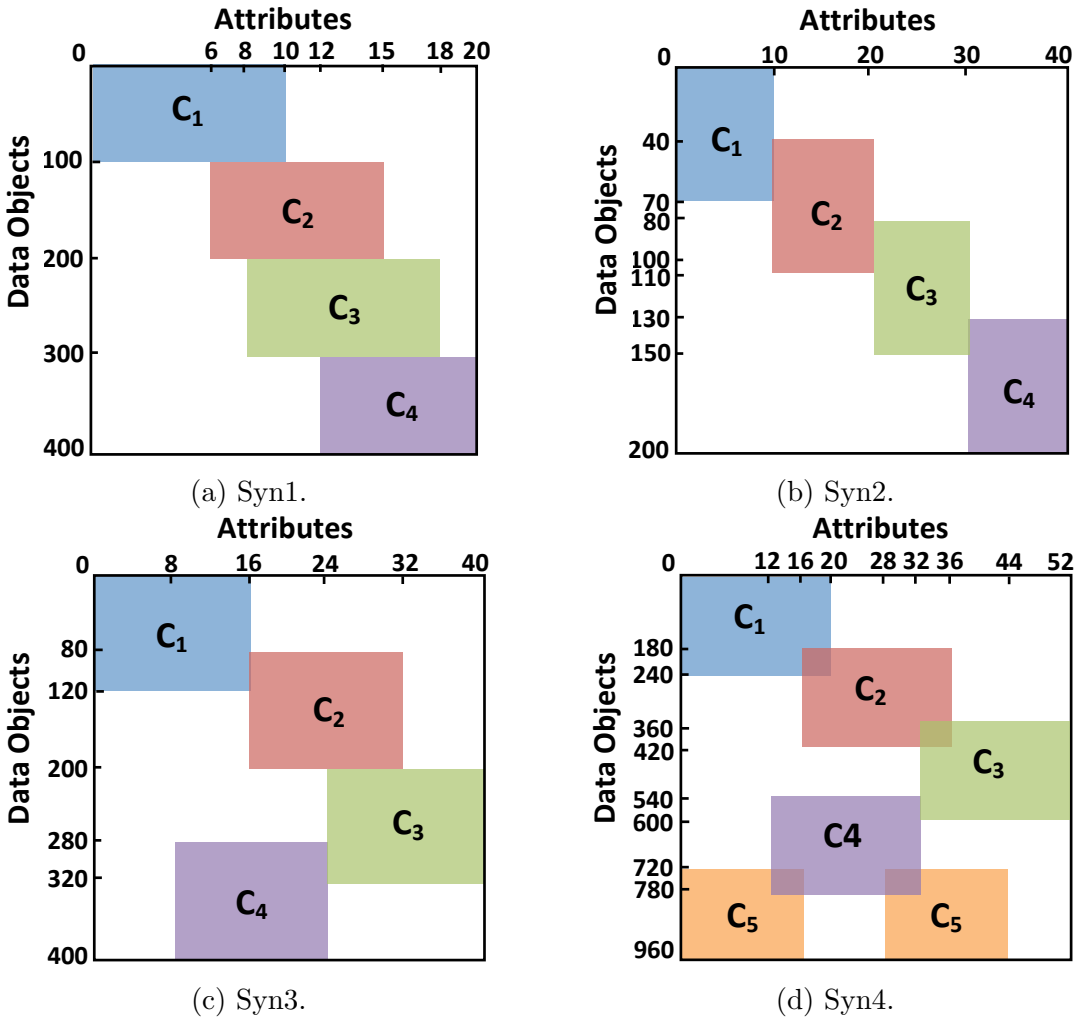
itemsets, we set it as for SUBCAD and Tiling. Two parameters are required for CLICKS (α and min_{sup}) and Hyper+ (false tolerant ratio f and min_{sup}) all from $[0, 1]$. We vary these parameters from 0.1 to 0.9 with a step of 0.1 for all data sets and report the best results. CLIQUE requires to pass grid size ξ and density τ as input parameters. We fix $\xi = 2 * W$ to fit categorical data, where W is the maximal number of categories. Then we vary τ from 0.1 to 0.9 with step 0.1 and report the best results. For Tiling, MTV and Hyper+, the points sets that support the detected itemsets might not cover all the points, thus we regard the rest as outliers like ROCAT.

To evaluate the cluster and subspace quality, we compare pairwise Precision, Recall and F-Measure as introduced in overlapping clustering literature [15, 47] for all data sets. A pair of points sharing at least one cluster is regarded as test outcome positive in clustering results or condition positive in golden standard. Precision is calculated as $\frac{t_p}{t_p+f_p}$ and Recall is obtained by $\frac{t_p}{t_p+f_n}$, where t_p, f_p and f_n are the numbers of true positives, false positives and false negatives respectively. F-Measure is the harmonic mean of Precision and Recall. In addition, we use the confusion matrix and the cluster content to evaluate the quality of all clusters and subspaces for the used real world data.

All experiments are performed on a workstation with 2.9 GHz Intel Core i7 CPU and 8.0 GB RAM.

3.4.1 Synthetic Data

We generate 4 synthetic data sets with different characteristics as depicted in Figure 3.4. $Syn1$ contains only overlapping attributes sets, whereas $Syn2$ contains only overlapping points sets. $Syn3$ adheres both kinds of overlapping, while $Syn4$ provides a more difficult scenario. The data sets are generated by first creating Pure Subspace Clusters and then randomly choosing 10% entries of each sub-matrix and randomly changing their values. Afterwards we randomly generate values for the remaining non-clustered area. The number of categories for each attribute is randomly chosen where the average number is 4. For each scenario we generate 5 data sets and report the average performance.



(a) Syn1.

(b) Syn2.

(c) Syn3.

(d) Syn4.

Figure 3.4: Synthetic categorical data for subspace clustering.

Table 3.1: Cluster Quality for Synthetic Data (F-Measure).

	Syn1	Syn2	Syn3	Syn4
ROCAT	**0.982**	**0.985**	**0.998**	**0.997**
SUBCAD	0.953	0.603	0.798	0.761
CLICKS	0.499	0.604	0.508	0.489
CLIQUE	0.414	0.604	0.507	0.516
DHCC	0.794	0.826	0.856	0.768
AT-DC	0.895	0.794	0.818	0.704
Tiling	0.542	0.532	-	-
MTV	0.509	0.496	0.439	0.526
Hyper+	0.565	0.634	0.565	0.491

Cluster Quality. Table 3.1 summarizes the results. Due to space limitations, the following part presents the F-Measure results only. ROCAT is the only algorithm performing very well on all the synthetic data sets with a F-Measure above 0.982. Note that these results are obtained without requiring any input parameters from the user. Designed for categorical subspace clustering, with suitable parametrization SUBCAD performs well on data set $Syn1$ containing clusters overlapping in the attributes (F-Measure 0.95). However, the performance of SUBCAD severely degrades if clusters overlap in the objects ($Syn2$, F-Measure 0.6). CLICKS and CLIQUE perform worse than ROCAT with a F-Measure of about 0.5, since they output too many redundant clusters (thousands or tens of thousands clusters). DHCC and AT-DC perform fairly well on all the data sets with a F-Measure of about 0.8. However, DHCC and AT-DC are limited to find full-dimensional clusters and therefore do not provide any information about the subspaces in which clusters are contained. Besides, they only find partitioned clusters without any overlapping information. The three informative itemset mining methods Tiling, MTV and Hyper+ do not perform well on our synthetic data sets either and yield F-Measures of about 0.5. The results of Tiling on $Syn3$ and $Syn4$ are discarded since the running time is over 1 hour.

Subspace Quality. In contrast to traditional clustering, subspace clustering does not only aim at finding clusters but also at identifying the subspaces containing clusters with high accuracy. Table 3.2 shows that ROCAT is the only technique correctly identifying the subspaces in all cases with a F-measure of 1. We discard DHCC and AT-DC, since they

Table 3.2: Subspace Quality for Synthetic Data (F-Measure).

	Syn1	Syn2	Syn3	Syn4
ROCAT	**1**	**1**	**1**	**1**
SUBCAD	0.975	0.524	0.967	0.949
CLICKS	0.742	0.375	0.375	0.528
CLIQUE	0.414	0	0	0
Tiling	0.808	0.849	-	-
MTV	0.831	0.777	0.638	0.621
Hyper+	0.565	0.469	0.744	0.853

do not support subspace clustering. SUBCAD performs well only if there is some overlap in the attributes ($Syn1$, $Syn3$ and $Syn4$), but the performance severely degrades on $Syn2$ where we only have overlap in terms of objects with an F-Measure of only 0.52. CLICKS and CLIQUE perform worst because they output too many redundant clusters. CLIQUE yields results with pairwise F-Measure values of 0 on $Syn2$, $Syn3$ and $Syn4$ because it outputs subspaces with a single attribute only. Tiling and MTV perform fairly well in terms of detecting subspaces with a F-Measure of 0.82 and 0.72 respectively. However, they only find the subsets of golden standard attributes sets. Hyper+ performs better with the more difficult scenarios $Syn3$ and $Syn4$ (F-Measure of about 0.8), but worse with the easier scenarios $Syn1$ and $Syn2$ (F-Measure of about 0.5). Since $Syn1$ and $Syn2$ are relative sparse, Hyper+ outputs more redundant clusters.

Robustness against outliers. We add different amounts of noisy objects to each synthetic data set. Particularly, we add 10% new records with random values in all attributes to $Syn1$ forming the noisy data $Syn1 - 10\%$ and analogously obtain other noisy data with different amounts. The pairwise F-Measure results are shown in Figure 3.5. We use the same settings for each scenario and for all the algorithms. Obviously ROCAT is extremely robust against noises. We cannot observe any decline in performance on $Syn3$ and $Syn4$. Moreover the decline is also negligible on the other two data sets yielding F-Measures above 0.96 on all examples even in the presence of 40% outliers. All the other algorithms severely degrade in performance in the presence of outliers, since they do not support the detection of noisy objects during the clustering process.

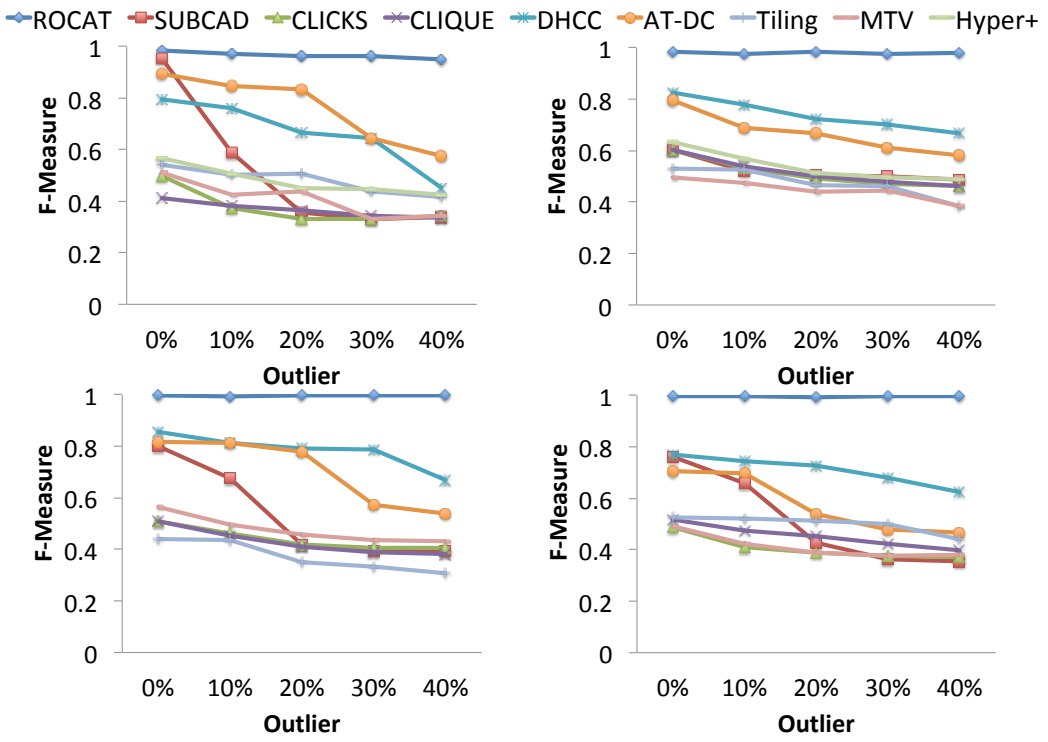


Figure 3.5: Robustness against outliers, *syn*1 to *syn*4 with outliers from left to right.

Scalability. To evaluate the scalability of ROCAT with respect to data size and dimensionality, we generate data sets using scenario 4 in Figure 3.4. For data size, each data set contains 52 attributes and the number of objects is varied from 10000 to 50000. For dimensionality, each data set contains 960 points and the dimensionality is varied from 50 to 200. The parameter settings are the same as for *Syn*4. Figure 3.6 summarizes the results. Some results are discarded if the running time is longer than 1 hour, i.e. SUBCAD and Tiling regarding data size and CLICKS and Tiling in terms of dimensionality. Figure 3.6 depicts that all the methods scale linearly in terms of number of objects. ROCAT performs similarly as DHCC and Hyper+, which is faster than MTV and slower than AT-DC, CLIQUE and CLICKS. With respect to dimensionality, ROCAT scales similar as DHCC, faster than SUBCAD and slower than AT-DC. CLIQUE, MTV and Hyper+ scale worst for dimensionality, where the running time severely increases when the dimensionality is added to 150 or 200.

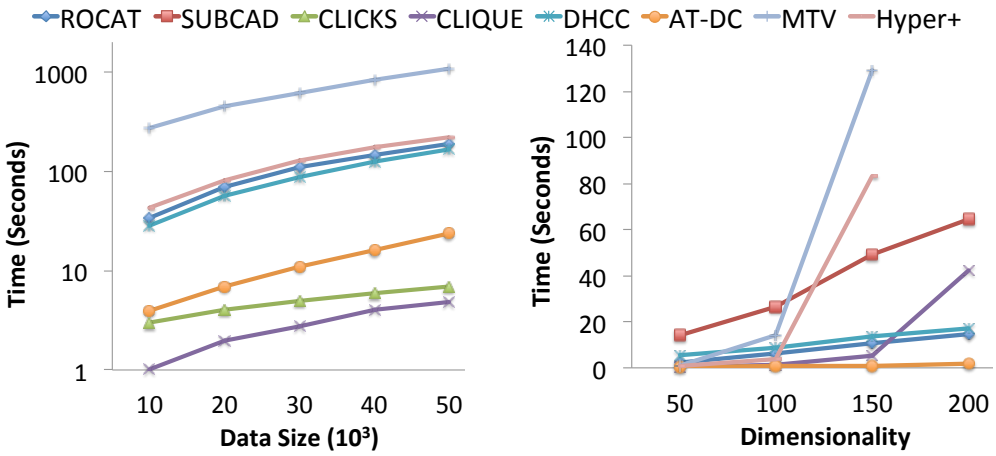


Figure 3.6: Scalability of ROCAT and comparisons.

3.4.2 Real World Data

In this section, we evaluate the performance of ROCAT and comparison methods on three real-world data sets: Congressional Votes, Mushroom and Molecular Biology (Splice-junction Gene Sequences) Data Set, which are publicly available at the UCI machine learning repository [1]. For these data sets, only non-overlapping class labels are available and there are only 2 or 3 classes. Moreover, most of the algorithms output many subspace clusters, which are normally the subsets of golden clusters. Therefore, Recall can not manifest the cluster quality anymore and we only use Precision for evaluation. We try different settings for all required parameters and choose the one with the best Precision as it is done for synthetic data sets. The results for real data sets are depicted in Table 3.3.

Congressional Votes. The data set consists of 435 instances, represented by 16 categorical attributes. There are 2 classes: democrat and republican. ROCAT and DHCC automatically output 2 clusters, while AT-DC finds 5 clusters. SUBCAD, Tiling and MTV output 2 clusters. Additionally, ROCAT, Tiling and MTV find an outlier cluster. CLICKS outputs 39 clusters, CLIQUE gives 12 clusters and Hyper+ provides 114 clusters. From Table 3.3 we can see that ROCAT outputs better clusters than most of the other methods with a Precision of 0.812. The confusion matrices are depicted in Table 3.4. Due to space

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

Table 3.3: Cluster Quality for Real Data (Precision).

	Vote	Mushroom	Splice
ROCAT	**0.812**	**0.999**	**0.861**
SUBCAD	**0.845**	0.501	0.378
CLICKS	0.525	0.508	0.343
CLIQUE	0.545	0.501	0.384
DHCC	0.793	0.766	**0.875**
AT-DC	0.521	0.612	0.497
Tiling	0.681	-	-
MTV	0.626	**0.943**	0.754
Hyper+	0.753	0.624	0.384

limitation, we only show the results of the top 6 methods and clusters with large number of points for those with too many clusters. Clusters with high purity are highlighted in bold.

ROCAT yields two clusters with very high purity, see. Table 3.4a. Regarding subspace quality, the clusters found by ROCAT are more compact in the detected subspace ($Cp = 0.194$) than in the whole space ($Cp = 0.254$) and the whole data set ($Cp = 0.531$). The compactness value $Cp \in [0, 1]$ is defined in [48], and 0 means that all data values in the corresponding features are the same. Specifically, let us take a look at cluster 0 in Table 3.4a, which is a pure democrat cluster. ROCAT outputs 12 attributes as the subspace for this cluster. More than 95% of voters in this cluster have the same opinion in 5 of the 12 subspace attributes, they voted *yes* to *aid to nicaraguan contras*, *yes* to *adoption of the budget resolution*, *no* to *physician fee freeze*, *no* to *el salvador aid*, and *yes* to *anti satellite test ban*. Further, at least 80% of the people vote for the same in the other 5 attributes, while more than 70% of them have the same vote in the final two attributes. We get similar statistics for the other cluster. Therefore, ROCAT does find meaningful subspaces for the detected clusters. Since SUBCAD also performs very well on this data set, let us take a look at its democrat cluster as well (cluster 1 in Table 3.4e). The corresponding subspace consists of 3 attributes only, which represents much less information. ROCAT is able to detect higher dimensional subspace clusters due to the ability to label objects as outliers. In detail, the votes of the instances labeled as outliers are nearly averagely distributed

Table 3.4: Results on Congressional Votes.

(a) ROCAT.

Cluster	Democrat	Republican
0	**148**	**0**
1	**20**	**136**
Noise	99	32

(b) DHCC.

Cluster	Democrat	Republican
0	49	159
1	**218**	**9**

(c) Hyper+.

Cluster	Democrat	Republican
0	**9**	**106**
1	**107**	**2**
92	28	20

(d) MTV.

Cluster	Democrat	Republican
0	**1**	**79**
1	**89**	**0**
Noise	177	80

(e) SUBCAD.

Cluster	Democrat	Republican
0	**27**	**154**
1	**240**	**14**

(f) Tiling.

Cluster	Democrat	Republican
0	**2**	**87**
1	**124**	**0**
Noise	141	81

in these 10 attributes. Therefore the properties of the outlier points are very different from those of the subspace clusters and thus it makes sense that ROCAT considers them as outliers. Tiling and MTV found outlier clusters as well. However, they can only find smaller Pure Subspace Clusters, which results in too many outliers. Some of the outliers that share similar subspaces as clusters are not detected.

Mushroom. The Mushroom data set contains 8124 records and 22 categorical attributes. Each record describes a mushroom specimen regarding 22 properties (e.g. shape, color, size) and is identified as definitely edible (4208 records) or poisonous (3916 records). ROCAT, DHCC and AT-DC automatically output 21, 10 and 6 clusters respectively. SUBCAD and MTV output 10 clusters. CLICKS outputs 260 clusters, CLIQUE gives 151 clusters and Hyper+ provides 183 clusters. Table 3.3 shows that ROCAT greatly outperforms the other methods with a Precision of 0.999. The confusion matrices of the top 6 methods are shown in Table 3.5. Not class-pure clusters are highlighted in bold.

Table 3.5a clearly shows that nearly all clusters detected by ROCAT are of high purity, which is much better than the other methods. Cluster 15 is the only one that contains several differently labeled records. However, DHCC, AT-DC and Hyper+ output clusters

Table 3.5: Results on Mushroom.

(a) ROCAT.

Cluster	Edible	Poisonous
0	1728	0
1	0	1728
2	0	1296
3	512	0
4	192	0
5	0	256
6	768	0
7	96	0
8	0	192
9	0	288
10	192	0
11	288	0
12	192	0
13	96	0
14	0	72
15	**48**	**32**
16	48	0
17	48	0
18	0	8
19	0	8
20	0	36

(b) MTV.

Cluster	Edible	Poisonous
0	0	1728
1	1	1296
2	1728	0
3	768	0
4	**288**	**192**
5	512	0
6	480	0
7	528	0
8	**384**	**48**
9	**496**	**224**
Noise	**192**	**524**

(c) CLICKS.

Cluster	Edible	Poisonous
43	1728	0
56	0	1728
201	**2516**	**80**
210	**4016**	**3856**
230	**2016**	**8**

(d) AT-DC.

Cluster	Edible	Poisonous
0	0	192
1	**798**	**1223**
2	**62**	**1065**
3	1296	0
4	1760	0
5	0	1728

(e) DHCC.

Cluster	Edible	Poisonous
0	**2880**	**736**
1	**808**	**72**
2	0	1296
3	216	0
4	0	1728
5	**32**	**24**
6	**64**	**16**
7	192	0
8	0	44
9	16	0

(f) Hyper+.

Cluster	Edible	Poisonous
1	1152	0
2	16	1296
3	0	1152
5	**855**	**353**
10	**1056**	**240**

with hundreds of misclassified objects, like Cluster 0 in Table 3.5e, cluster 1 in Table 3.5d and cluster 5 in Table 3.5f. MTV is the second best algorithm on Mushroom data regarding Precision, because most of the clusters are Pure Subspace Clusters. However, there are still many clusters with hundreds of misclassified objects, like cluster 9 in Table 3.5b. CLICKS outputs many high purity clusters, however, some clusters contain 50 percent misclassified records, like cluster 210 in Table 3.5c. Besides, cluster 210 shows that the overlapping clusters provided by CLICKS are highly redundant. Cluster 210 contains nearly all the points and only one attribute as the subspace. In contrast ROCAT also finds overlapping clusters in the searching phase, however the redundancy is removed during the Combining and Reassigning phase. Finally, ROCAT outputs the most relevant subspace clusters without any redundancy.

In terms of subspace quality, the compactnesses Cp of the clusters found by ROCAT are 0.126 in the detected subspace, 0.171 in the whole space and 0.518 in the whole data set. Let us take Cluster 2 in Table 3.5a as an example. The 1296 mushrooms in this cluster are all poisonous. The subspace that this cluster exists in is composed of 14 attributes. Specifically, the cluster consists of specimen *without bruises* and *foul odor*, *free close broad gill*, with identical *shape*, *root* and *surface* of stalk, *partial white veil*, *one large ring* and the color of spore is *chocolate*. Mushrooms with these features are all poisonous. To validate the relevance of this subspace attributes, we calculate the category distribution of the remaining 8 attributes. The result indicates that the mushrooms in these attributes have different category values. For example, the *cap-shape* attribute, contains one half *bell* shaped and the other half *flat* records. Moreover, the *gill-color* attribute exhibits *buff*, *chocolate* and *green* mushrooms. Similar statistics can also be found on other clusters. Consequently, ROCAT can not only detect clusters, but can find the subspaces as well.

Splice. This data set consists of 3190 instances and 60 categorical attributes. The instances are gene sequences and attributes are the positions on the sequences. The value of each attribute is a DNA base (A, T, G, C). Splice contains class labels designating instances as either EI (767 records), IE (768 records) or Neither (1655 records). EI and IE denote that exon/intron boundaries and intron/exon boundaries can be recognized in the

sequence, respectively. Neither states that there are neither EI nor IE sites.

Table 3.6: Results on Splice.

(a) ROCAT.

Cluster	EI	IE	Neth.
0	**45**	**703**	**16**
1	**629**	**16**	**10**
2	1	17	0
3	0	8	0
4	15	0	0
5	9	0	0
6	8	0	0
7	10	0	0
Noise	**101**	**36**	**1629**

(b) AT-DC.

Cluster	EI	IE	Neth.
0	55	78	513
1	151	644	1058
2	**561**	**46**	**84**

(c) Hyper+.

Cluster	EI	IE	Neth.
0	**374**	**8**	**3**
52	153	75	127
197	300	302	709
349	130	4	442

(d) MTV.

Cluster	EI	IE	Neth.
0	**489**	**7**	**1**
1	175	605	38
2	294	50	10
3	28	170	6
4	35	142	5
Noise	**130**	**96**	**1601**

(e) CLIQUE.

Cluster	EI	IE	Neth.
0	174	158	380
95	161	197	378
179	227	123	402
240	136	153	393

(f) DHCC.

Cluster	EI	IE	Neth.
0	15	10	419
1	**668**	**19**	**28**
2	40	3	393
3	11	0	369
4	**28**	**728**	**34**
5	5	8	412

ROCAT, DHCC and AT-DC automatically output 8, 6, and 3 clusters respectively. Besides, ROCAT identifies 1,766 points as outliers. SUBCAD and MTV output 5 clusters. CLICKS, CLIQUE and Hyper+ output 256, 241 and 399 clusters respectively. From Table 3.3 we can see that ROCAT outperforms most of the other methods with a Precision of 0.861. The confusion matrices of top 6 methods are shown in Table 3.6. Clusters with good quality regarding the number of contained points and purity are highlighted in bold.

Table 3.6a clearly illustrates that the clusters found by ROCAT are of very high purity. Cluster 0 and Cluster 1 contain the majority of all data points and are very pure. Cluster 0 is composed of 92% objects from class *IE* and Cluster 1 contains 97% objects from class *EI*.

Besides, the outliers detected by ROCAT are mainly composed of records in the *Neither* class. DHCC and MTV also perform well on Spice. The resulting clusters are very pure as well, like clusters 1 and 4 in Table 3.6f and cluster 0 and the noise cluster in Table 3.6d. Although DHCC finds many clusters that mainly contain records of the *Neither* class, it cannot label them as outliers. On the other hand, MTV is able to find noisy cluster, but also finds clusters of lower purity compared to ROCAT. The other methods do not perform well on Splice.

The compactnesses Cp of the clusters found by ROCAT are 0.249 in the detected subspace, 0.371 in the whole space and 0.741 in the whole data set, which indicates the good quality of detected subspaces. Particularly, we choose cluster 0 of ROCAT in Table 3.6a as an example to show the effectiveness of ROCAT on detecting subspaces. The subspace is made up of 25 out of the 60 original attributes. Among the detected 25 attributes, there are 2 positions (28 and 29) with the same values (A and G) for all sequences. Moreover 90 percent of the genes include C on position 27. Besides, there are 10 and 12 positions where more than 80 and 60 percent of all genes only take 2 different base values, respectively. On the other hand, the 4 categories $\{A, T, G, C\}$ are averagely distributed in the remaining 35 positions by nearly all the gene sequences in Cluster 0. Therefore, ROCAT outputs reasonable subspaces for the Splice data set.

CLICKS, CLIQUE and Hyper+ output overlapping clusters on Splice data set. However, there are too many clusters with a large amount of redundancies. It is hard for users to interpret such results directly. The other methods all provide partition-based results. In contrast, ROCAT finds relevant overlapping subspace clusters on Splice data set. There are 63 objects with multiple labels and 5 pairs of clusters sharing objects. Cluster 1 and 4 in Table 3.6a for example, share 15 records. However, they are detected in different subspaces: 6 attributes for cluster 1 and the other 52 attributes for cluster 4. Cluster 1 is very compact in the 6 detected attributes, while the 15 instances in cluster 4 are also very similar in further 52 attributes. Therefore, the overlapping clusters provide additional information over other partition-based algorithms. Further, ROCAT only provides the most relevant clusters without any redundancy, which facilitates the interpretation of

the clustering results.

3.5 Related Work and Discussion

3.5.1 Categorical Subspace Clustering

Compared to the large body of literature on clustering numerical data only relatively few papers focus on clustering categorical data. Some prominent approaches include the basic algorithm K-modes [65] extending the famous K-means algorithm to categorical data, ROCK [55] and COOLCAT [16], to mention a few. It is often difficult to find clusters in the full dimensional space even in moderate-dimensional data sets, and a problem that is known as the *curse of dimensionality* has been extensively studied. For an comprehensive survey on clustering high-dimensional numerical data see [72]. One of the most prominent technique is CLIQUE [10]. This grid-based approach actually discretized the numerical data and therefore is also applicable to categorical data. However, it enumerates all the possible subspace clusters which produces large redundancies.

Less algorithms have been designed for categorical subspace clustering. Ganti et al. [49] proposed the categorical clustering method CACTUS, which builds a summary information from the data set first and then projects the cluster onto each attribute. It can be extended to find subspace clusters, however though introduced in the paper, it was not implemented by the authors [116]. Gan and Wu [48] proposed the categorical subspace clustering algorithm SUBCAD. They define a cost function based on the idea that data points in relevant subspaces are compact while being sparse in irrelevant ones. LIMBO [13] is a hierarchical algorithm based on an information bottleneck framework. They try to maximise the mutual information between the clusters and attribute values. A good cluster accurately predicts the attribute values associated with objects of the cluster. Although LIMBO is based on information theories it does - in contrast to ROACT - not take into account the model complexity. Furthermore, CACTUS, SUBCAD and LIMBO are all partition-based method, which cannot find overlapping clusters, and need input parame-

ters. CLICKS [116] is a subspace algorithm which constructs a k-partite graph based on all the values of all attributes and then searches for maximum cliques. CLICKS supports overlapping clustering, however, it often includes too many redundant clusters. Besides, the input parameters are hard to determine without having deeper knowledge of the data.

Subspace clustering methods are either partition-based or produce too many redundant clusters. To solve the redundant problem, STATPC [87] and RESCU [88] are proposed to find relevant non-redundant subspace clusters in high-dimensional numerical data. However, they are not applicable for categorical data. Moreover, they need many parameters to bound the searching space.

Only very few algorithms support parameter-free clustering of categorical data. Xiong et al. [114] proposes a divisive hierarchical algorithm DHCC, which iteratively splits the higher level cluster by Multiple Correspondence Analysis (MCA) and then refines the result. Cesario et al. [28] proposes a top-down algorithm AT-DC, which iteratively generates and stabilizes clusters to achieve best quality. DHCC and AT-DC are both parameter-free methods based on a top-down splitting framework, thus they can only find partitioning clusters but not overlapping clusters. Besides, DHCC and AT-DC are greatly affected by outliers, where ROCAT can handle them very well.

3.5.2 Informative Pattern Mining

Pattern mining is another area related to the problem of categorical subspace clustering. For instance, the frequent itemsets found by pattern mining methods could be regarded as the subspaces of clusters, while the objects that support the itemsets can be seen as clusters. Among these pattern mining algorithms, informative itemset mining, which finds the most informative itemsets or ranks the importance of the itemsets, is the most relative one. For instance, Tiling [52] defines a tile as a region in the 0/1 database where all values are 1 (Subspace Cluster), which aims at finding a tiling consisting of at most K tiles covering the largest possible area. Tiling can only find tiles without fault-tolerance, besides it needs the number of tiles as input parameter. NoisyTile [70] uses the maximum entropy distribution

to measure the informativeness of a tile or tiling and it supports noisy tiles. However, it needs a fault-tolerant itemset mining algorithm, i.e. [94], to generate the candidate noisy tiles. Moreover it only gives a rank of informativeness on itemsets. Similarly Hyper+ [113] tries to find overlapped hyper-rectangles (noisy tiles) from candidates that are generated by an itemset mining method with a different cost function. KRIMP [108] and MTV [85] are designed for informative itemset mining based on compression. KRIMP needs a minimum support value as input parameter while MTV is parameter-free. However, they do not support fault-tolerant itemsets thereby performing worse than ROCAT in our experiments. The cost function of ROCAT is different and thus their searching or ranking methods cannot be directly applied. Besides, ROCAT is fully automatic while most of these algorithms need input parameters. Furthermore, ROCAT scales better than these algorithms in terms of both data size and dimensionality.

3.6 Conclusion

In this chapter, we introduced ROCAT, an effective and efficient algorithm for detecting the most relevant overlapping subspace clusters on categorical data. Combining a compression-based view on clustering with an effective search algorithm, ROCAT identifies the truly relevant subspace clusters which may overlap in terms of the assigned objects and/or the constituting attributes. The compression-based approach of ROCAT naturally avoids undesired redundancy of the result and guarantees that each detected cluster is relevant since it contributes to compress the data. In ongoing and future work, we explore extending our idea to support numerical data or mixed-type data.

Chapter 4

Multiple Subspace Selection for Hierarchical Clustering

The last chapter focuses on improving the effectiveness of subspace clustering. Another challenge for clustering high-dimensional data is the interpretation of the results. For data sets with correlated and irrelevant features, meaningful clusters often exist in different arbitrarily-oriented subspaces. Such situation makes the interpretation even more difficult, since the detected subspaces are arbitrarily-oriented and there is no semantic meaning for them. In literature, only very few approaches touch the challenge of interpreting the arbitrarily-oriented subspace clusters.

In this chapter, we try to handle the interpretation challenge by proposing a clustering framework named MSS that provides hierarchical clustering and visualization. Parts of the material presented in this chapter have been submitted in [61], where Xiao He was mostly responsible for the development of the main concept, implemented the main algorithms and wrote the largest parts of the paper; Claudia Plant supervised the project and proposed the initial idea of multiple subspace visualization; Sebastian Goebl and Son T.Mai helped with the implementation and performed parts of experiments; Christian Böhm revised the whole paper; The co-authors also contributed to the conceptual development and paper writing.

"Xiao He, Sebastian Goebl, Son T.Mai, Christian Böhm and Claudia Plant. Multiple Subspace Selection for Hierarchical Clustering and Visualization. Submitted for publication."

MSS integrates Orthogonal Linear Discriminant Analysis, K-means and Kernel Density Estimation techniques. It detects multiple low-dimensional subspaces, each exhibiting an interesting cluster structure. Moreover, our technique includes an intuitive visualization of the relevant subspaces by scatter plots showing the cluster structure together with histograms showing the contribution of the original features. Further facilitating the interpretation, MSS discovers a hierarchy of clusters in multiple subspaces. Our algorithm is robust and provides high-quality clusters even in the presence of a large number features not relevant for clustering. Extensive experiments on both synthetic and real data sets show the effectiveness and efficiency of MSS compared to existing methods.

The remainder of this chapter is organized as follows: In Section 4.1, it starts with an introduction. We briefly give some backgrounds about LDA and orthogonal LDA in Section 4.2. Section 4.3 presents the algorithm MSS in detail. Section 4.4 contains an extensive experimental evaluation. Section 4.5 briefly discusses related work and Section 4.6 concludes the chapter.

4.1 Introduction

In many applications ranging from market segmentation to biomedicine, clustering can offer deep insights into the major trends in our data without requiring much previous knowledge. Moreover, clustering commonly is the first step in a data mining workflow to explore an unknown data set. Most existing clustering approaches mainly aim at detecting high-quality clusters. However, the outputs of many techniques tend to be difficult to interpret. In the following steps of the data mining workflow, the interpretation of a clustering result certainly is as important as the quality of the detected clusters. Due to the presence of irrelevant or correlated features clustering of already moderate-dimensional

data is a challenging task since meaningful clusters often exist in different arbitrarily-oriented subspaces. This phenomenon does not only make clustering difficult, but also the interpretation of the clustering result.

In literature, arbitrarily-oriented subspace clustering techniques have been proposed to detect such clusters, e.g. ORCLUS [9], 4C [23] and COPAC [4]. ORCLUS combines PCA and K-means to achieve arbitrarily-oriented clusters, while 4C and COPAC integrate PCA into density-based clustering method. These methods assume each cluster being located in an unique arbitrarily-oriented subspace. The results are hard to interpret, since the detected subspaces are arbitrarily-oriented and there is no relationship among the clusters. Specifically, these techniques provide no knowledge of in which subspace two particular clusters can be well separated.

The technique LDA-Km [40] approaches the challenge in a different way: This approach integrates the supervised dimensionality reduction technique Linear Discriminant Analysis (LDA) with K-means into a subspace clustering framework, which finds a single subspace for all clusters. The greatest benefit of single subspace is the possibility to project all data to a joint space. If the joint space is low-dimensional, we can easily visualize the data by scatter plots. By inspecting this joint low-dimensional space we can learn a lot about the relationships among the clusters. However, in most cases there are many clusters in a data set, and a single two- or three dimensional subspace is not sufficient to distinguish all the clusters. In LDA-Km, the authors set the dimensionality of reduced subspace to $K - 1$, where K is the number of clusters, which makes visualization impossible when the number of clusters exceeds four.

Observed from many real datasets, we found that a single low-dimensional subspace is not sufficient to distinguish all clusters, while each single cluster in its individual subspace cannot give important relationships between each other. Rather, a group of clusters can be well distinguished in one subspace and another group of clusters in a different one. Therefore, we aim at finding a set of low-dimensional arbitrarily-oriented subspaces, which contain interesting cluster structure. Specifically, we propose a top-down splitting framework based on Orthogonal Linear Discriminant Analysis for detecting subspace clusters

in multiple low-dimensional subspaces. In each detected subspace, there are multiple pronounced clusters and their relationships can be explored by inspecting the corresponding subset of the data in the subspace. The generated cluster hierarchy further facilitates interpretation. For simplicity sometimes we use 'subspace' instead of 'arbitrarily-oriented subspace' in the following.

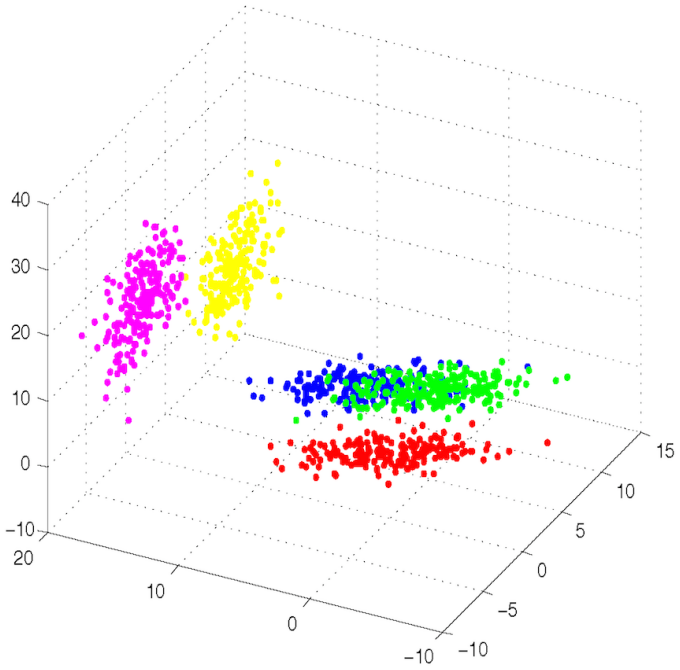


Figure 4.1: An example data set for illustrating the idea of hierarchical visualization.

We use a simple example to illustrate our idea, which is depicted in Figure 4.1. The data set, with 1000 points and 3 dimensions, contains 5 clusters. There are two groups of clusters, each sharing a similar orientation: The first group is formed by the purple and yellow clusters and the second group by the remaining clusters (red, blue and green). Obviously, it is not possible to transform this data set into a 2-dimensional subspace where all 5 clusters can be well distinguished. Our aim is to detect hierarchical clusters which exist in multiple subspaces. Then we can interpret them by a hierarchical visualization, which is shown in Figure 4.2. At first, the original data is projected to a two-dimensional subspace shown as *Subspace*1 in Figure 4.2, where two clusters are well separated. Then, the original data of each cluster is projected to the other two subspaces shown as *Subspace*2

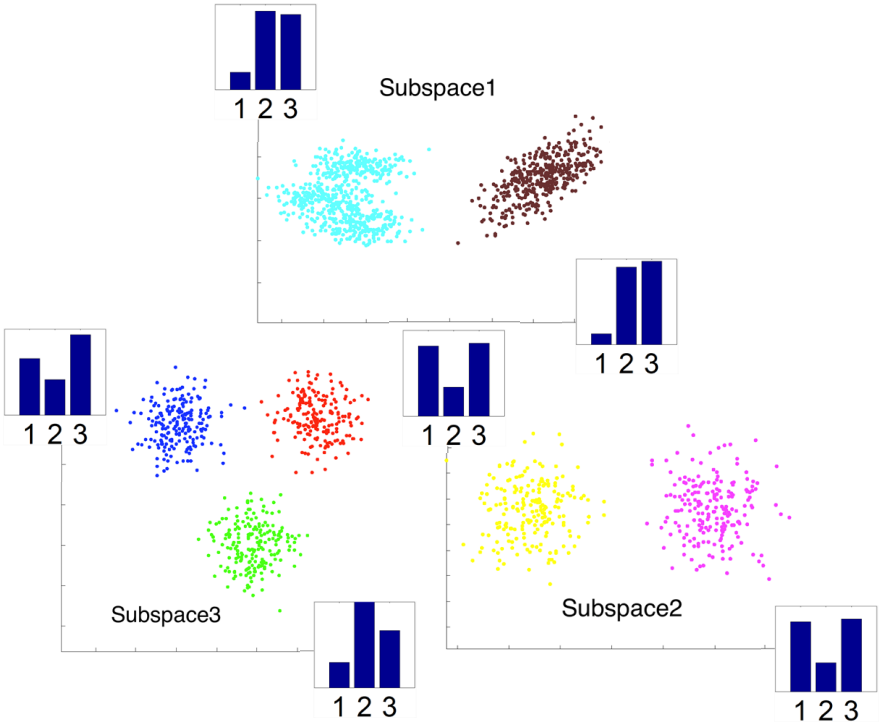


Figure 4.2: Hierarchical clustering and visualization.

and *Subspace*3 in Figure 4.2. In *Subspace*2, two lower level clusters can be found, while in *Subspace*3 three clusters can be well distinguished. From this we learn the relationship between subspace clusters. The blue bar charts in the axis are contribution histograms of original features. For instance, in *Subspace*1 original feature 2 and 3 contribute more on distinguishing blue and brown clusters, while in *Subspace*2 feature 1 and 3 contribute more on separating purple and yellow clusters. Furthermore, The difference between subspaces can be seen from the bar charts as well.

The main contributions of this chapter are:

1. **Multiple Subspace Selection.** We propose a framework MSS (Multiple Subspace Selection) that finds multiple low-dimensional subspaces, each exhibiting an interesting cluster structure.

2. **Hierarchical Visualization.** We provides hierarchical visualization with different low-dimensional subspaces for an intuitive interpretation of the clustering result.

3. **High Quality Clustering Results.** Our technique outperforms state-of-the-art comparison methods on multiple synthetic and real data sets in terms of clustering quality.

4. **No need of specifying parameter of subspace dimensionality.** Many methods need a parameter to specify the dimensionality of subspace, which is hard to estimate. Our technique avoids this by providing low-dimensional subspaces and hierarchical structure.

5. **Robust to irrelevant Dimensions.** Our method is very robust in the presence of a large number of irrelevant dimensions.

4.2 LDA and Orthogonal LDA

For a data set with K clusters, in Linear Discriminant Analysis (LDA) we use the between-class scatter and within-class scatter matrices:

$$S_b = \sum_k n_k (m_k - m)(m_k - m)^T \tag{4.1}$$

$$S_w = \sum_k \sum_{i \in C_k} (x_i - m_k)(x_i - m_k)^T \tag{4.2}$$

where m_k is the mean of class C_k and m is the global total mean.

The optimal subspace $U = U_1, ..., U_d$ is obtained by optimizing:

$$\max_U \mathrm{Tr} \frac{U^T S_b U}{U^T S_w U} \tag{4.3}$$

where d is the dimensionality of the subspace, which is usually set to $K - 1$.

In [82] D. Luo et al point out that the classic LDA is implicitly defined with constraints. The classic LDA is not orthogonal as PCA, *i.e.* $U^T U \neq I$. However, in many cases we desire that the projection directions are mutually orthonormal. Therefore, they propose orthogonal LDA optimizing Eq. (4.3) under the orthogonality constraint $U_{orth}^T U_{orth} = I$.

According to [82], the solution of U_{orth} for orthogonal LDA is obtained as follows. Firstly, we get the principal eigenvectors $F = \{f_1, ..., f_d\}$ (associated with d largest eigenvalues) of $S_w^{-\frac{1}{2}} S_b S_w^{-\frac{1}{2}}$. Then the solution of S_w-orthonormal LDA (with the constraint $U_w^T S_w U_w = I$) is:

$$U_w = S_w^{-\frac{1}{2}} F. \tag{4.4}$$

We have automatically $U_w^T S_w U_w = I$, because $S_w^{-\frac{1}{2}} S_b S_w^{-\frac{1}{2}}$ is a positive definite symmetric matrix, and $F^T F = I$. Finally, the solution of orthogonal LDA is:

$$U_{orth} = U_w (U_w^T U_w)^{-\frac{1}{2}} \tag{4.5}$$

With orthogonal LDA, the transformation does not change the scale of the original data, which is important for interpretation of the clustering result: All the resulting subspaces during the run of our algorithm correspond to orthonormal rotations followed by projection and we do not want to modify the data in any other way like scaling or warping. Thus, we know that the resulting cluster structure is present in a subspace of the original data set.

4.3 Multiple Subspace Selection

For a given data set, our aim is to find a group of transformations that project the data to different low-dimensional subspaces $U = \{U_1, U_2, ..., U_p\}$. In each subspace U_i, a group of clusters $C_i = \{C_{i1}, C_{i2}, ..., C_{iq}\}$ can be well clustered, where $i \in \{1, 2, ..., p\}$. By doing this, we can interpret the clustering results by visualizing them in different low-dimensional subspaces. The problem is very difficult to solve since we do not know how many subspaces we need and how many clusters are in each subspace.

In this chapter, we use a hierarchical framework to model this problem: To allow for visualization and interpretation, we restrict ourselves to 2-dimensional subspaces in the following. If desired e.g. by application-specific needs, our framework can also be modified to detect higher dimensional subspaces. We assume that any data set with cluster structure can be transformed to an arbitrarily-oriented subspace in which at least 2 clusters can be

Algorithm 3 OLDA-Km

Input: K, $Data$
Output: $Clusters$, $TransData$

Set dimensionality of subspace $d = 2$;
$TransData$ = PCA($Data$, d);
while Not converge **do**
  $Clusters$ = KMeans($TransData$, K);
  $TransData$ = OLDA($Data$, $Clusters$, d);
end while

return $Clusters$, $TransData$;

well separated. Obviously, a one-dimensional subspace is able to separate 2 clusters. If rotating the data set and projecting it to all possible one-dimensional subspaces cannot partition the data set into at least 2 clusters, the dataset must be a cluster itself. The higher level clusters are separately projected to the other arbitrary subspaces, and their members can be well split into clusters there. Finally, we build the hierarchy of the whole data set.

4.3.1 Orthogonal LDA-Kmeans

One important step of our framework is to transform the original data set to a 2-dimensional subspace where we can detect at least two clusters in it. This reminds us about the technique LDA-Kmeans (LDA-Km) [40] that integrates the supervised method into the clustering process to find clusters in an arbitrarily-oriented subspace. However, the original LDA-Km projects the data to $K - 1$ dimensions and changes the scale of original data, which is, as discussed before not desirable for visualization. Therefore, we use Orthogonal LDA and fix the projected dimensionality with 2 for visualizing data in the original scale. Additionally we find that Orthogonal LDA-Kmeans (OLDA-Km) usually performs a superior than LDA-Km in our experiments. The pseudocode code of OLDA-Km is depicted in Algorithm 3.

Does OLDA-Km always converge? The answer is yes. Similar as LDA-km [40], the

objective function of OLDA-km is the same as of OLDA. There are two steps in OLDA-km: the K-means in the subspace U and OLDA in the original space. Obviously, the second step OLDA monotonically maximize the objective function. And the objective function of OLDA Eq. (4.3) is equivalent to:

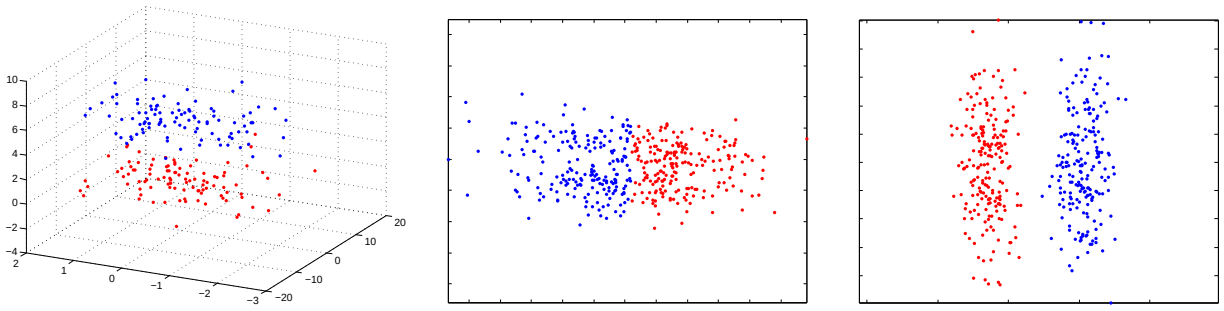
$$\min_{U} \mathrm{Tr} U^T S_w U \tag{4.6}$$

since the covariance matrix $S_t = S_w + S_b$ is constant. Therefore the objective of K-means in subspace U is equivalent to OLDA's objective function, as Eq. (4.7) shown. Therefore, the objective function is monotonically maximized in this step as well. Finally OLDA-Km converges to a local optimum, since each step they increase the objective function of Eq. (4.3).

$$\begin{aligned} \min_{C} \mathrm{Tr} U^T S_w U &= \sum_k \sum_{i \in C_k} \parallel U^T x_i - U^T m_k \parallel^2 \\ &= \mathrm{Tr} \sum_k \sum_{i \in C_k} U^T (x_i - m_k)(x_i - m_k)^T U \end{aligned} \tag{4.7}$$

OLDA-Km aims to find the directions U that maximize the between-class distances and minimize the within-class distances in the transformed subspace. OLDA needs the correct clustering to find the subspace that best distinguishes clusters, while K-means performs better when the data is projected to the best subspace. OLDA-Km solves this chicken-egg problem iteratively by performing transformation and clustering to get the best clusters and subspace simultaneously. However, since K-means is heavily dependent on the initialization and only find the local optimum, similarly OLDA-km converges to local optimum and relies on the initialization as well.

To acquire stable clustering results, we try to avoid this issue in OLDA-Km. The simplest way would be performing OLDA-Km multiple times and choose the best result based on the objective function of OLDA Eq. (4.3). However, only considering the distances between different clusters centers and the distances with-in cluster do not work all the time



(a) Original 3D data

(b) Incorrect clustering with Eq. (4.3) = 1.56 and Eq. (4.8) = 0.08

(c) Correct clustering with Eq. (4.3) = 0.81 and Eq. (4.8) = 1.34

Figure 4.3: The different objective function values on a 3D example data.

for subspace clustering. Because usually there are heavy correlations among the features. For example there are two 3-dimensional clusters, which are very close from each other and their projection is shown in Figure 4.3. Their features are heavily correlated and their intrinsic dimensionality are 2. LDA-Km and OLDA-Km always converge to bad results when setting dimensionality of subspace $d = 1$. For $d = 2$ LDA-Km and OLDA-Km output different results from diverse initializations. Unfortunately Figure 4.3b provides incorrect results with a higher objective function value Eq. (4.3) of 1.56, while Figure 4.3c gives the correct clustering with a lower objective value of Eq. (4.3) is 0.81.

It is necessary to consider the separation among clusters while choosing the best local optimum results, see Figure 4.3. Therefore, we modify Eq. (4.3) by adding a weight shown in Eq. (4.8), which is the single-link distance between different clusters in transformed subspace. Single-link distance between two clusters is the shortest distance between their objects as shown in Eq. (4.8). We use single-link distance but not complete-link or average-link distance. The reason is that complete-link and average-link distance have the similar effects as the objective function of OLDA, which take all the objects of clusters into consideration. By adding the single-link distance, we consider the separateness among different clusters. For the example in Figure 4.3, the new heuristic function gives 0.08 to the correct clustering in Figure 4.3b and 1.34 to the incorrect situation in Figure 4.3c. In our experiments, we found that executing OLDA-Km 20 times would be a good choice to

get stable results based on Eq. (4.8).

$$\begin{aligned} \max_{U}(\frac{\mathbf{Tr}U^T S_b U}{\mathbf{Tr}U^T S_w U} \cdot \min_{C_i, C_j \in C} Dist(C_i, C_j)) \\ Dist(C_i, C_j) = \min_{x \in C_i, y \in C_j} dist_{D \cdot U}(x, y) \end{aligned} \tag{4.8}$$

where $dist_{D \cdot U}(x, y)$ means the distance between object x and y in the subspace U.

4.3.2 Multiple Subspace Selection

LDA-Km reduces the dimensionality of data to $d = K - 1$, where K clusters can be found there. However, in practice the number of clusters K is usually larger than expected. Therefore, the found arbitrarily-oriented subspace owns a higher dimensionality as well, which makes visualization impossible when K is larger than 4. Further, the performance of K-means also deteriorates when K increases, which affects the effectiveness of LDA-Km and OLDA-Km.

We bring a top-down splitting algorithm for multiple subspace selection that make better clustering and interpretation of the results. Our idea is based on the assumption that any dataset with cluster structure can be transformed to an arbitrarily-oriented subspace, in which at least 2 clusters can be well separated. Specifically, we greedily split the cluster (initial cluster contains all data points) in an arbitrarily-oriented subspace with OLDA-Km until no cluster can be well partitioned. Kernel density estimation is used to estimate the densities of points in subspace, and the difference between densities of cluster centre and border are compared for termination.

The pseudocode of our algorithm MSS is provided in Algorithm 4. At first, we try to partition all the points in a subspace with OLDA-Km, then each resulting cluster in a lower hierarchy is partitioned in the other subspace. In each step, OLDA-Km is performed on the objects in the original space with $k \in [2, ..., MaxK]$. $Maxk$ is a parameter that indicate the maximum number of clusters can be found in a single subspace. From our experiments, we found that there is no difference when $MaxK > 5$ in a wide range. Therefore, $MaxK$

Algorithm 4 MSS

Input: ϵ, $MaxK$, $Data$
Output: $Subspaces$, $Hierarchy$

Add all objects to partition queue $Queue$;
while $Queue$ is not empty **do**
  $Objs = Queue$.Pop;
  Get the original data $Odata$ of $Objs$;
  for $k \in [2, ..., MaxK]$ **do**
    $[Clusters, Tdata]$ = **OLDA-Kmeans**(k, $Odata$);
    if Eq. (4.11) $> \epsilon$ in two closest clusters **then**
      Break;
    else
      Keep $Clusters$ and $Tdata$ with maximum Eq. (4.8);
    end if
  end for
  $Queue$.AddAll($Clusters$);
  $Subspaces$.Add($Clusters$, $Tdata$);
  Update $Hierarchy$ with $Clusters$;
end while

return $Subspaces$, $Hierarchy$;

is set to 5 in all the following experiments. Kernel density estimation is used to judge whether the resulting clusters are well-separated. The one with biggest objective function of OLDA-Km in Eq. (4.8) will be kept and split later. Finally, the splitting is terminated when no leaf cluster can be partitioned any more and the hierarchy is built in process. The real parameter for MSS is ϵ, which is a threshold to evaluate the difference between densities of cluster center and border.

The multivariate kernel density estimation is defined as follows:

$$\hat{f}(x) = \frac{1}{N} \sum_{y \in D} (\prod_{i=1}^{d} \frac{1}{h_i} F(\frac{x - y}{h})) \tag{4.9}$$

where $h = h_1, ..., h_d$ is the bandwidth and the term $F(\cdot)$ is a d-dimensional kernel function that is non-negative and integrates to one. We use the standard multivariate normal kernel:

$F(x) = (2\pi)^{-d/2}exp(-x^2/2)$. The bandwidth h is selected using an established heuristic which is proven to work well in various applications [102]:

$$h_i = (4/(d+2))^{\frac{1}{d+4}} \cdot \sigma_i \cdot N^{-\frac{1}{d+4}} \tag{4.10}$$

where σ_i the standard deviation of the cluster.

To determine whether clusters are well-separated or not, at first, we find two closest clusters of the results from OLDA-Km. By closest clusters, we mean that the single-link distance between the two clusters is shortest in the transformed subspaces. Then we define the separateness for stopping the partition as Eq. (4.11) shows, where $maxDensity$ is the maximum density of the object inside the cluster, and P is the object of the cluster which is nearest to the other cluster, $densityP$ is the density of P.

$$Separateness = \frac{maxDensity - densityP}{maxDensity} \tag{4.11}$$

If $Separateness$ is bigger than a predefined threshold ϵ, we think the clusters are well-separated. From Eq. (4.11) we know that $Separateness \in [0, 1]$, therefore ϵ is set to $[0, 1]$. In the experiments section, we will study the effect of ϵ on the proposed algorithm MSS and show that MSS is quite stable with a wide range of ϵ choices.

4.3.3 Runtime Complexity

The runtime complexity of MSS for a data set with N points and D dimensions is equivalent to $MaxK \cdot t \cdot$ (OLDA-Km + SL), where $MaxK$ is a constance and fixed to 5 in MSS, t is the number of times that OLDA-Km performs, and SL is the complexity for computing Single-link distances, which is $O(N^2)$. Term t relies on the parameter ϵ and bigger ϵ leads to smaller t. Usually $t \ll N$. Therefore, the complexity of MSS is equivalent to (OLDA-Km + $O(N^2)$). The complexity for K-means can be approximated to $O(ND)$ since it always converge fast. For OLDA we need to make matrix multiplication and inversion for a D dimensional matrix. Theoretically the time complexity of this procedure

is $O(D^3)$. However, it can be calculated in $O(D^{2.375})$ by Coppersmith-Winograd algorithm [34]. Therefore, the complexity for OLDA-Km is $O(nND + nND^{2.375})$, where n is the number of iterations of OLDA-Km to converge, which is usually small. Finally, the runtime complexity of MSS can be approximated to $O(ND^{2.375} + N^2)$.

4.4 Experiments

This section provides empirical evidences to show the effectiveness of MSS on both synthetic and real data sets. After giving the experimental setup, we evaluate MSS in five aspects: Firstly, we compare OLDA-Km, one important component of MSS, with LDA-Km. Secondly, we study the effect of the only parameter ϵ in MSS and show its stability. Thirdly, we compare MSS with state-of-art arbitrarily-oriented subspace clustering methods in terms of clustering quality. After that, we show the hierarchical visualization produced by MSS with a case study. Finally, we evaluate the scalability of MSS.

4.4.1 Setup.

We Compare MSS against all arbitrarily-oriented subspace clustering algorithms implemented in the framework ELKI [6], namely ORCLUS [9], 4C [23] and COPAC [4]. In terms of detecting hierarchical structure in arbitrarily-oriented subspace, HiCO [5] is the first algorithm that address the problem. Therefore we show the comparison with HiCO as well. Furthermore, we compare MSS with LDA-Km [40], which integrates supervised dimensional reduction technique to subspace clustering. K-means is chosen as the baseline method as well. We implement K-means, MSS, LDA-Km and OLDA-Km in Java and get the others from ELKI package [1]. All experiments have been conducted on a workstation with 2.9GHz dual-core CPU and 8.0 GB RAM.

We generate 2 basic synthetic datasets $Clus5$ and $Clus10$ with hierarchies. $Clus5$ is a 5 dimensional dataset with 5 clusters, while $Clus10$ is a 10 dimensional dataset with 10

[1] http://elki.dbs.ifi.lmu.de/

clusters. Each cluster contains 200 points. The hierarchical structure of $Clus10$ is depicted in Figure 4.4, while $Clus5$'s structure is like the left child of root in Figure 4.4. Each square represents a cluster, the purple one is the root cluster containing all the points. Clusters with same color can be distinguished in an arbitrarily-oriented subspace. Moreover, leaf clusters with same color are correlated and share the same orientation, e.g. clusters in Figure 4.1. We generate a correlated cluster by first building a 3-dimensional Gaussian cluster with standard deviations 1.0, 1.0 and 4.0 in each dimension. Then we rotate it with a 3d orthonormal matrix. Leaf clusters with same color share the same rotation matrix. Furthermore, we add irrelevant dimensions to each group of clusters. Take the 3 yellow clusters in Figure 4.4 for example, we put them in the first 3 dimensions and add 7 irrelevant dimensions with uniform distribution ranging from [0,10], and put the following 2 red clusters in $3, 4, 5$ dimensions and add irrelevant dimensions analogously. In the same way, we generate the $Clus5$ dataset with a easier hierarchy.

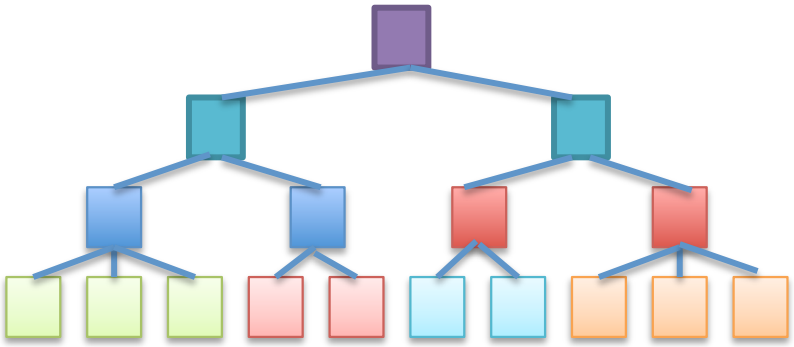


Figure 4.4: The hierarchical structure of dataset $Clus10$.

Six real-world data sets are chosen from UCI machine learning repository [2] for evaluation. They are *iris*, *wine*, Pen-based Recognition of Handwritten Digits (*Pendigits*), Image Segmentation (*Segment*), *Ecoli*, and Landsat Satellite (*Satellite*). The descriptions of these datasets are depicted in Table 4.1. All the synthetic and real data sets in Table 4.1 have labels. We view the labels of the data sets as the ground truth and use normalized mutual information (NMI) and F-Measure as performance measure.

[2] http://archive.ics.uci.edu/ml

Table 4.1: Datasets description.

Datasets	Samples	Dimensions	Class
Clus5	1000	5	5
Clus10	2000	50	10
Iris	150	4	3
Wine	178	13	3
Pendigits	7494	16	10
Segment	2310	19	7
Ecoli	336	7	8
Satellite	6435	36	6

Table 4.2: Parameterizations on real datasets.

	ORCLUS (k, l)	4C (ϵ, m, λ)	COPAC (ϵ, m, k)	HiCO (μ, k)
Clus5	$5, 5$	$3.1, 2, 5$	$2.5, 5, 2$	$30, 30$
Clus10	$10, 50$	$10, 10, 50$	$6, 2, 2$	-
Iris	$3, 4$	$1, 10, 4$	$10, 2, 12$	-
Wine	$3, 13$	$40, 5, 13$	$6, 8, 39$	-
Pendigits	$10, 16$	$21, 2, 8$	$13.4, 3, 48$	$5, 20$
Segment	$7, 19$	$20, 2, 10$	$5.5, 2, 57$	$20, 20$
Ecoli	$8, 7$	$0.1, 2, 5$	$0.02, 2, 21$	$25, 25$
Satellite	$6, 36$	$36, 15, 36$	$12, 2, 98$	-

The same with MSS, K-means and LDA-Km, we execute ORCLU 100 times for each dataset and show the mean of the NMI and F-Measure values. We give the true number of clusters to K-means, LDA-Km and ORCLUS as their input parameter. Besides, ORCLUS needs the average number of intrinsic dimensionality l for subspace clusters, we test l from $[1, .., D]$ and output the best result, where D is the dimensionality of data. 4C and COPAC need 3 parameters, we try ϵ and m (minpts) in a wide range ($[0.01, 40]$ and $[2, ..., 25]$ respectively), and set the third parameter following the author's instruction [23, 4]. Finally, we output the best results. HiCO needs four parameters, we use default δ and α in ELKI package and try different μ and k. The parameterizations for all comparison methods are shown in Table 4.2.

4.4.2 Comparing OLDA-Km with LDA-Km

OLDA-Km is one important component of MSS, in this part we show the improvement of OLDA-Km on LDA-Km on datasets with a small number of clusters ($Clus5$, $Iris$ and $Wine$), since no hierarchy can be found there. Besides, OLDA-Km with its objective function Eq. (4.3) named OLDA-KmO and the proposed heuristic function Eq. (4.8) OLDA-KmH are compared as well. We execute OLDA-Km 50 times and choose the results based on Eq. (4.3) and Eq. (4.8) for the results of OLDA-KmO and OLDA-KmH. The NMI and F-Measure results are summarized in Table 4.3 and Table 4.4. The performance of these algorithms depend on initialization, thus for each dataset we execute them 100 times and show the mean of the values.

Table 4.3: NMI between OLDA-Km and LDA-Km.

Algorithm	Clus5	Iris	Wine
K-means	0.693	0.676	0.383
LDA-km	0.703	0.886	0.634
OLDA-Km	0.741	**0.919**	0.727
OLDA-KmO	0.827	**0.919**	0.695
OLDA-KmH	**0.941**	**0.919**	**0.734**

Table 4.4: F-Measure between OLDA-Km and LDA-Km.

Algorithm	Clus5	Iris	Wine
K-means	0.645	0.673	0.459
LDA-km	0.668	0.937	0.733
OLDA-Km	0.701	**0.961**	0.793
OLDA-KmO	0.811	**0.961**	0.779
OLDA-KmH	**0.938**	**0.961**	**0.797**

From Table 4.3 and 4.4 we can clearly see that OLDA-Km improves the clustering results of LDA-Km with higher NMI and F-Measure values on all the three datasets by using the orthogonal transformation. OLDA-KmO with OLDA's objective function improves the clustering quality on synthetic dataset $Clus5$, but degrades the result on Wine dataset, which means that the objective function of OLDA is not always a good metric for subspace clusters. Further, OLDA-KmH produces the best results, which show the effectiveness of

proposed function Eq. (4.8). Besides, all the LDA-based method improve the clustering results of K-means.

4.4.3 The effect of parameter Epsilon

Before comparing MSS with the other competitors, we first evaluate the parameter ϵ of MSS. From Eq. (4.11) we know that $\epsilon \in [0, 1]$. In Figure 4.5, we report the clustering qualities (NMI) of MSS with ϵ ranging from 0.05 to 1 with step 0.05 on all data sets. As we can see, our algorithm is robust to the choices of ϵ in a wide range of choice. For example, the clustering quality of dataset *Pendigits* remains stable from 0.6 to 0.8 for 85% values of ϵ. Besides, in most cases we get the best results with $\epsilon \in [0.4, 0.6]$, which is our recommendation for using MSS. There are some exceptions, e.g. Iris. However, ϵ is still easy to set, because for a higher ϵ MSS outputs nothing, which is helpful to address the boundary.

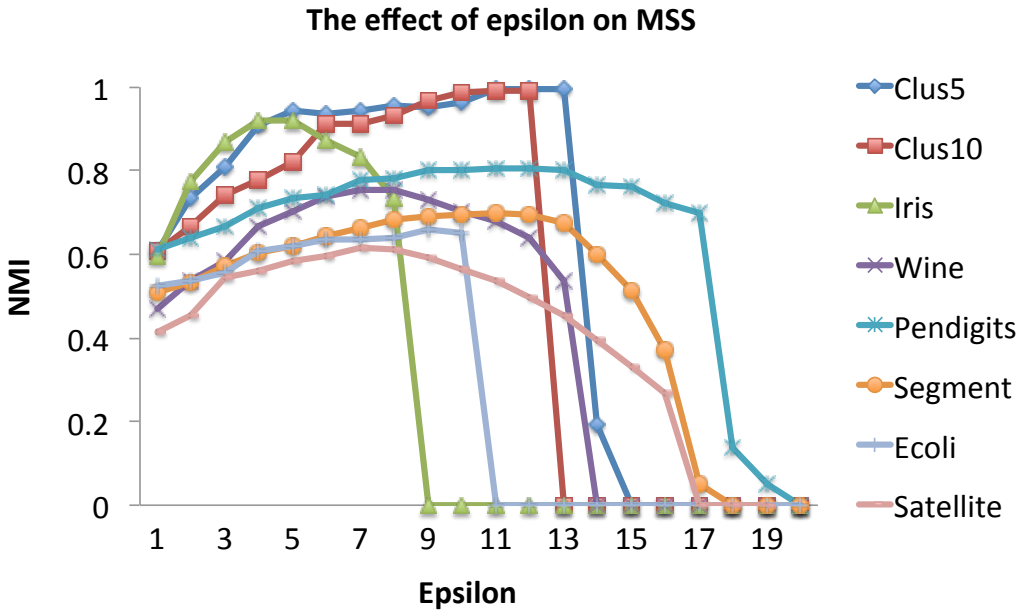


Figure 4.5: The effects of parameter ϵ on synthetic and real dataset for MSS.

4.4.4 Clustering Quality

In terms of clustering quality, we evaluate MSS in two aspects. First, we compare the NMI and F-Measure values on both synthetic and real datasets. Secondly, we evaluate the effects of irrelevant dimensions.

Table 4.5: Clustering quality comparison (NMI)

Algorithm	Clus5	Clus10	Iris	Wine	Pendigits	Segment	Ecoli	Satellite
MSS	**0.992**	**0.981**	**0.907**	**0.757**	**0.812**	**0.675**	**0.655**	**0.571**
K-means	0.693	0.697	0.678	0.383	0.692	0.524	0.583	**0.569**
LDA-Km	0.703	0.673	0.886	0.634	0.681	0.337	-	**0.578**
ORCLUS	0.836	0.801	0.756	0.394	0.678	0.408	0.631	0.446
4C	0.893	0.618	0.733	0.389	0.477	0.574	0.393	0.482
COPAC	0.759	0.644	0.046	0.355	0.664	0.555	0.206	0.291
HiCO	0.456	-	-	-	0.271	0.272	0.369	-

Table 4.6: Clustering quality comparison (F-Measure)

Algorithm	Clus5	Clus10	Iris	Wine	Pendigits	Segment	Ecoli	Satellite
MSS	**0.991**	**0.965**	**0.948**	**0.819**	**0.756**	**0.575**	**0.787**	**0.561**
K-means	0.645	0.514	0.673	0.459	0.616	0.462	0.518	**0.569**
LDA-Km	0.668	0.499	0.937	0.733	0.595	0.341	-	**0.575**
ORCLUS	0.825	0.693	0.806	0.591	0.576	0.401	0.597	0.485
4C	0.922	0.415	0.746	0.577	0.214	0.444	0.441	0.501
COPAC	0.787	0.535	0.032	0.588	0.585	0.478	0.454	0.375
HiCO	0.353	-	-	-	0.254	0.212	0.336	-

Table 4.5 and 4.6 summarize the clustering results. LDA-Km does not work on Ecoli due to the singular problem. Hico cannot provide any useful clusters for some data sets as well. It is clear that MSS outperforms all the comparison methods in all the tested data sets. Specifically, MSS provides nearly perfect results for two synthetic datasets $Clus5$ and $Clus10$. Besides, MSS outputs good NMI and F-Measure values for $Iris$, $Wine$, $Pendigits$ and $Segmentation$ as well, which are at least 0.12 higher than those from comparisons. In terms of the other two data sets, MSS and ORCLUS perform equally good on $Ecoli$, while MSS, K-means and LDA-Km produce similar result on $Satellite$. Besides MSS works better than the other methods.

Real world data sets often contain irrelevant features. To test the robustness of MSS against irrelevant features, we add different amount of irrelevant dimensions to dataset *Clus*5. Specifically, we add additional dimensions with uniform distribution ranging from $[0, 10]$. The results of MSS and comparison methods are shown in Figure 4.6. For 4C, COPAC and HiCO, the original parameters for *Clus*5 do not work, we try different parameterizations and output the best one in each case. For the other methods, we use the original parameters for *Clus*5 including MSS. From Figure 4.6 it is evident that MSS is extremely robust against irrelevant dimensions. The decline is very minor with a NMI above 0.96 even with 5 data and 25 irrelevant dimensions. All the other algorithms severely degrade the performance in the presence of irrelevant dimensions. 4C and COPAC are affected more by irrelevant dimensions.

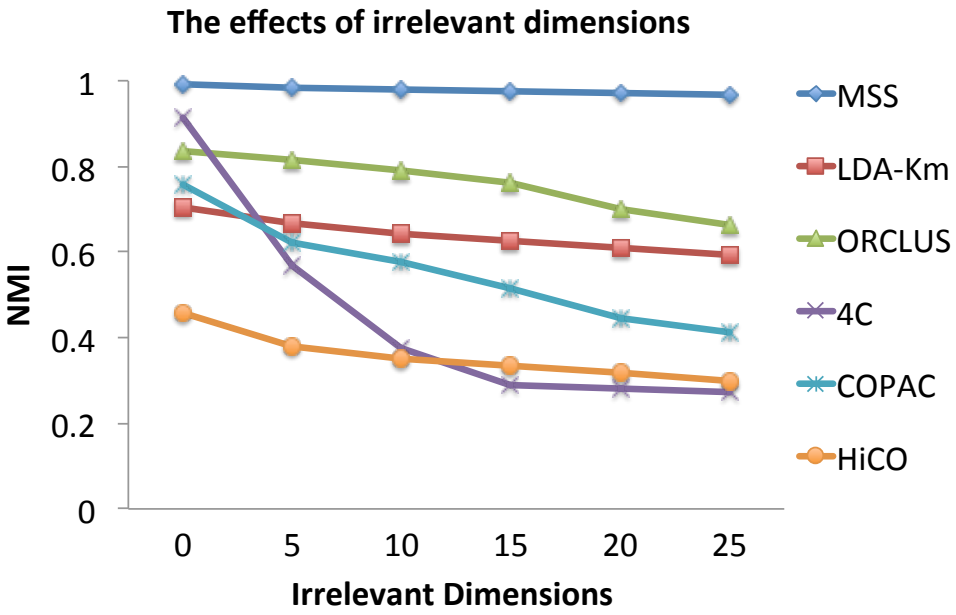


Figure 4.6: The effects of irrelevant dimensions.

4.4.5 A Case Study on Pendigits Data

In this section, we give a case study on Pendigits data set for hierarchical clustering and visualization in multiple subspaces provided by MSS. Pendigits data set contains 7494

points and 16 dimensional features of hand-written digits from 30 writers. The objects are labeled according to the digit.

The hierarchical structure computed by MSS is depicted in Figure 4.7. The root cluster (purple block) contains all the digits, the blue blocks represent the intermediate hierarchical clusters while the yellow blocks are leaf clusters. Children clusters from the same father can be well separated in an arbitrarily-oriented subspace, some of them are labeled in the figure. The digits in each block mean that the cluster contains these digits. The percentage following the digit in leaf cluster represents the proportion of the digit the cluster owns.

From Figure 4.7, it is clear that the clustering accuracy is very high. Nearly all the leaf clusters are dominated by a single digit. Specifically, there are 13 leaf clusters in total and 8 clusters among them are consisted of more than 94% of 7 different digits. Besides, the other 3 are composed of 2 similar digits and one of them forms the majority (more than 75%), e.g. $\{1, 7\}$ and $\{2, 1\}$. Furthermore, there are 2 leaf clusters in the lower hierarchy composed of two similar digits with analogous percentages. And they only own a small number of points compared to the others. Apart from the accuracy, MSS detects interesting hierarchy from *Pendigits* data set as well. Take the descendants from the left child of root cluster for example, at first odd digits and even digits are separated in *Subspace*3, then the even digits are further split into $\{0, 8\}$ and $\{4, 6\}$ in *Subspace*4. Clearly, 0 and 8 are similar as well as 4 and 6. Finally, they are distinguished in *Subspace*9 and *Subspace*8. The detected hierarchy is just as that in real life.

With the hierarchical structure and the transformed data in multiple subspaces produced by MSS, we build the hierarchical visualization for further interpreting the clustering results. Due to space limitation the hierarchical visualization for 7 2-dimensional subspaces (the part inside the red line of Figure 4.7) are shown in Figure 4.8. Clusters are labeled with different colors in each subspace. The arrow goes from the father cluster to a subspace, where the father cluster can be well-separated. Besides, the true label of each instance is shown as the digit itself as well. From Figure 4.8 we can see that different digits can be distinguished in different subspaces. Particularly, in *Subspace*4, *Subspace*8 and *Subspace*9 where only leaf cluster exists, same digits are close in the corresponding subspace while

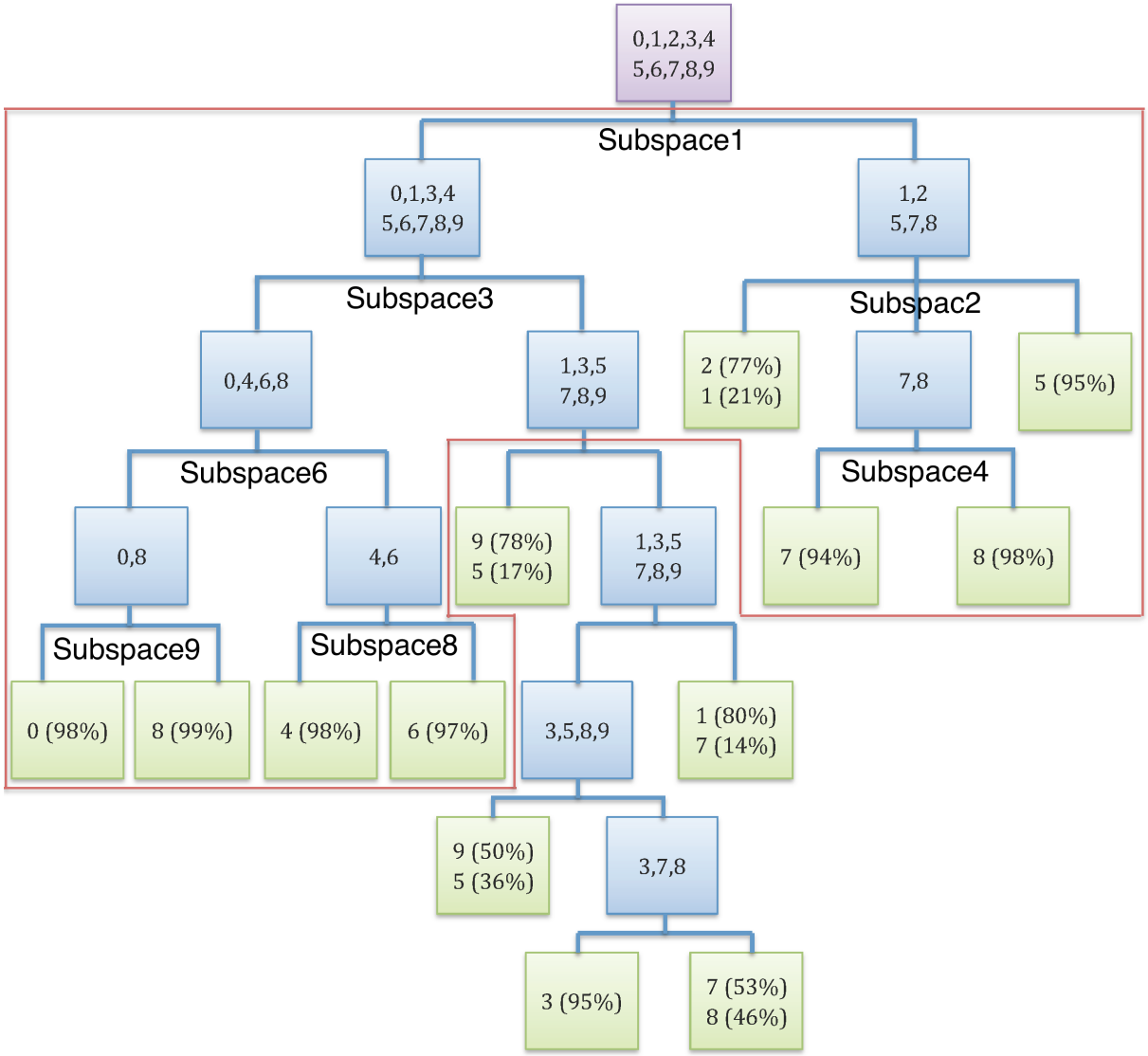


Figure 4.7: Hierarchy of Pendigits detected by MSS.

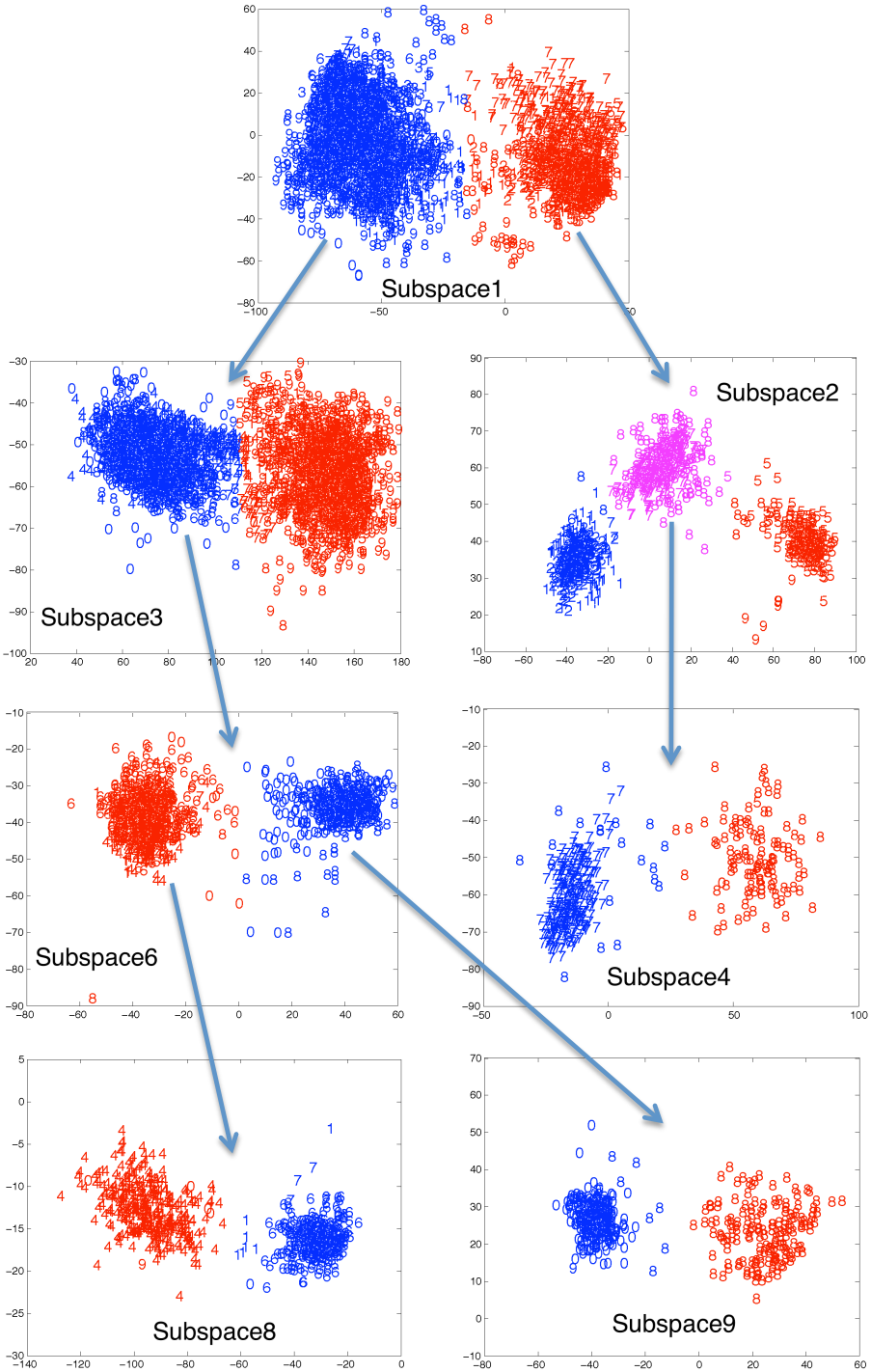


Figure 4.8: Hierarchical visualization on subspaces of Pendigits dataset found by MSS.

different digits are far from each other. The subspaces found by MSS are effective for separating the corresponding clusters.

MSS provides transformation matrix for each resulting subspace. We further analyze the contribution of original features for the subspaces in Figure 4.8. We show the contribution histograms for 4 subspaces only with leaf clusters in Figure 4.10. We give the contribution histograms of horizontal axis since these leaf clusters can be well-clustered there. From Figure 4.10a, we can easily get that Feature 1, 5 contribute most for distinguishing digits 7 and 8 in *Subspace*4. While separating digits 0 and 8 in *Subspace*9, feature 6 and 9 contribute most, which is depicted in figure 4.10d.

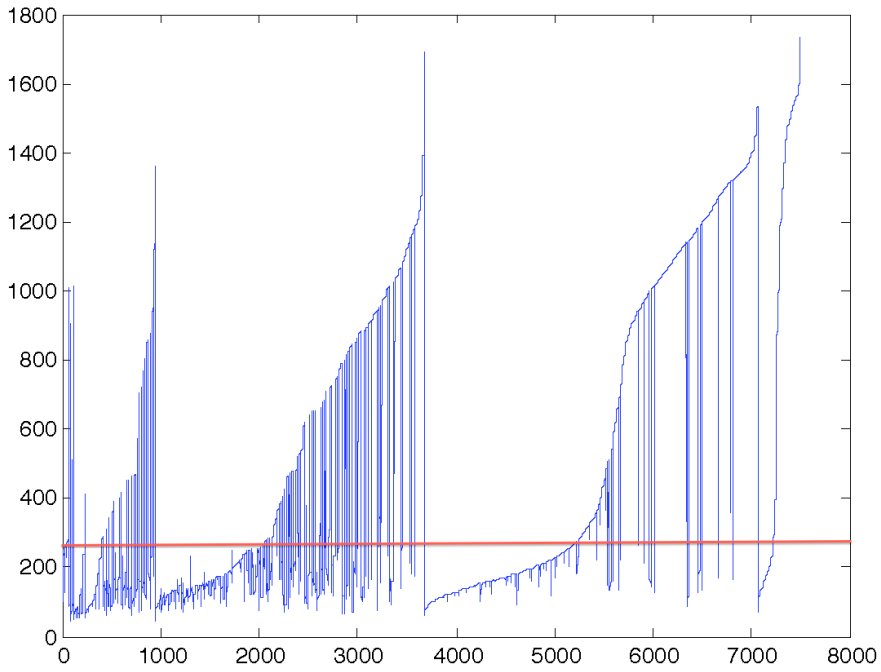
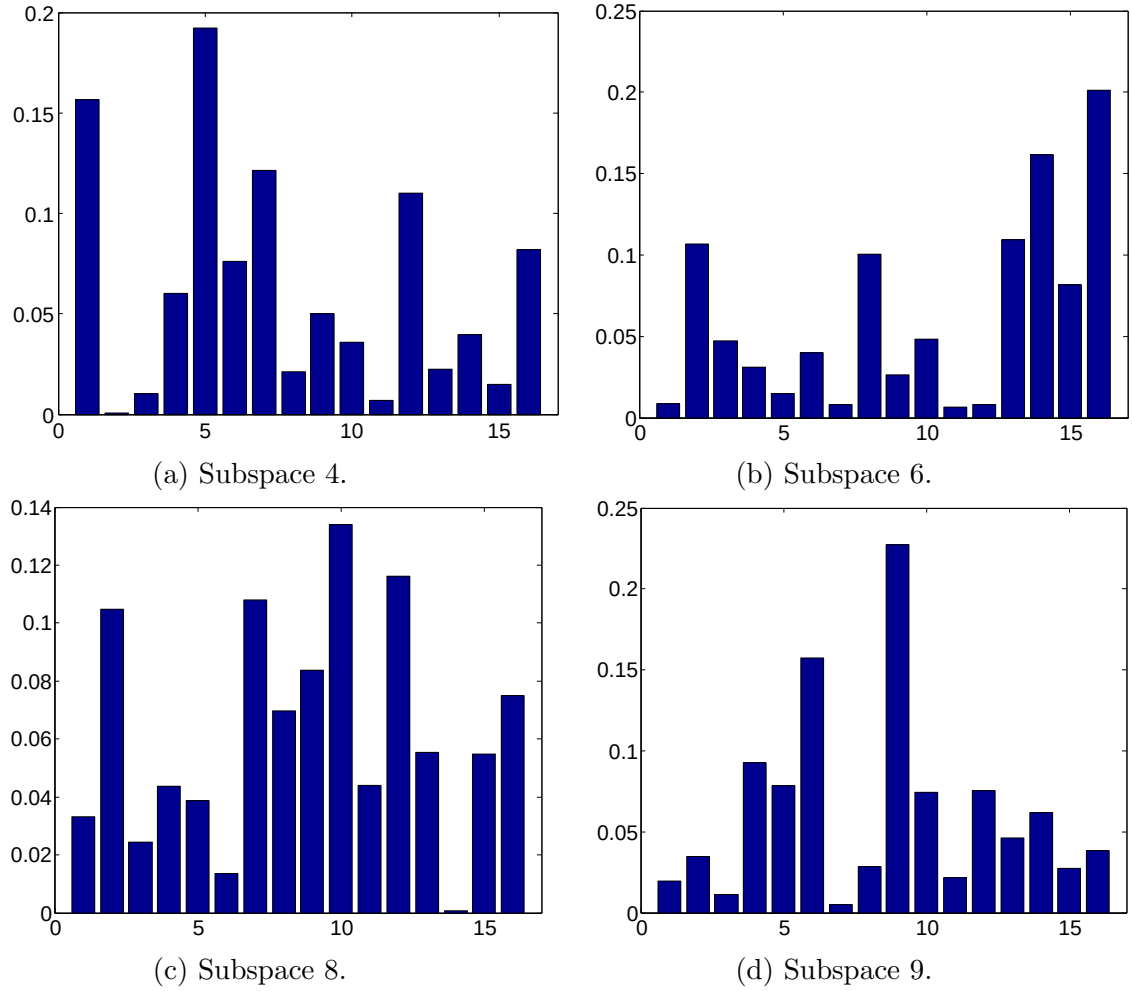


Figure 4.9: Reachability plot from HiCO on Pendigits with $\mu = 5$ and $k = 20$, NMI=0.271.

HiCO outputs points order and their reach-abilities like OPTICS [14] as its clustering results. For example, the best result of HiCO on *Pendigits* data set with $\mu = 5$ and $k = 20$ is shown in Figure 4.9. The ordering of the points is on the x-axis and the reachability distance is on the y-axis. We get the clusters by cutting at the red line with the reachability of 245.1, and the points in the same valley form a cluster. The result of HiCO on Pendigits is much worse than that of MSS and MSS provides more information by the proposed



(a) Subspace 4. (b) Subspace 6.

(c) Subspace 8. (d) Subspace 9.

Figure 4.10: Contribution histograms of original features for subspaces in Figure 4.8.

hierarchical visualization.

From the hierarchical visualization and the contribution histogram, we can learn more about the relationship between clusters, e.g. in which subspace they can be best separated, which features contribute more to the subspace, how close are these clusters in the subspace, are there some instances in the border line and so on.

4.4.6 Scalability

To evaluate the scalability of MSS with respect to data size and dimensionality we generate datasets using the way of creating $Clus5$. For data size, each data set contains 5 dimensions

and varying number of points from 2000 to 10000. For dimensionality, each data set contains 1000 points and varying the dimensionality from 10 to 50. Figure 4.11 and Figure 4.12 summarizes the results. MSS scale similarly as most comparison methods in the number of objects, which is faster than the other hierarchical subspace method HiCO. With respect to dimensionality, MSS scales similar with ORCLUS and LDA-Km as we analyzed before. 4C and COPAC scale better in the dimensionality with correct parameter settings. However, searching for a good parameterization is a difficult task as well. HiCO performs worst and is not scalable in dimensionality.

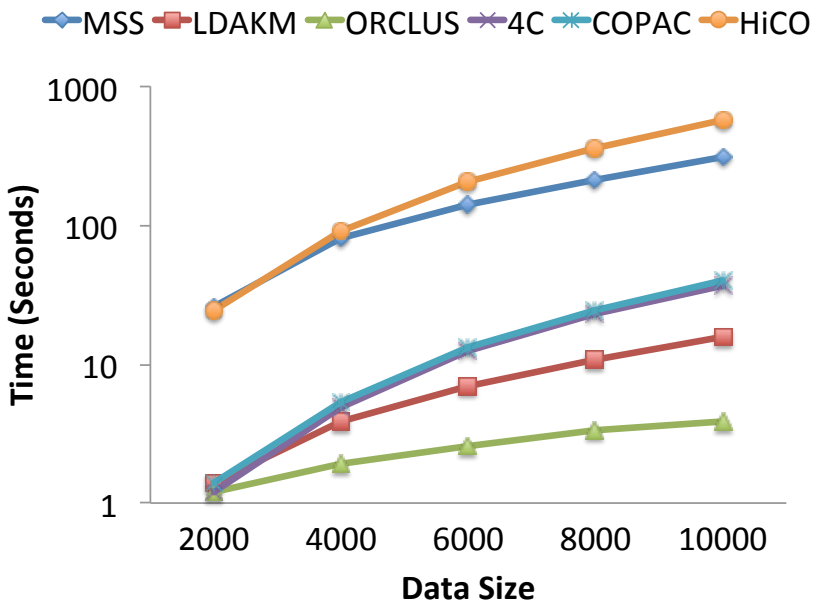


Figure 4.11: The scalability of MSS and comparisons regarding Data size.

4.5 Related Work and Discussion

4.5.1 Exploiting Supervised Techniques for Clustering

Clustering and classification are closely related tasks and sometimes clustering can be considered as unsupervised classification. Classification can profit from clustering, e.g. clustering inside the classes has been demonstrated to improve classification accuracy [46]

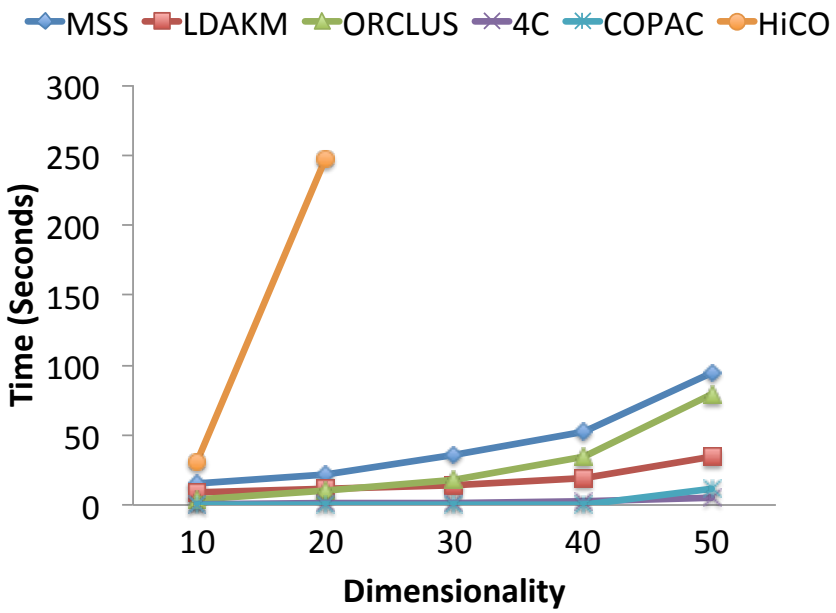


Figure 4.12: The scalability of MSS and comparisons regarding dimensionality.

and is very helpful to accelerate the training of support vector machines [25]. More related to our work are papers exploiting classification techniques for clustering. There are several papers on support vector clustering e.g. [18, 76], to mention a few. These methods rely on the idea that first mapping data to a higher-dimensional feature space, where a minimal sphere can be found. Then separate clusters can be found when mapping the sphere back to data space. However, they can not handle moderate ot high dimensional data, which is not suitable for our problem. Most related to our work is the approach LDA-Km [40] that integrate Linear Discriminant Analysis into clustering. LDA-Km iteratively do LDA and K-means and finally the data are clustered while the subspace is selected simultaneously. We also base our technique MSS on LDA because we can project clusters to a same subspace and learn their relationship. But instead of classic LDA we use orthogonal LDA [82]. In contrast to standard LDA, this orthonormal transformation corresponds to a rotation of the data space but does not alter the original scale and variance of the data. Further, our framework MSS select multiple subspaces and build hierarchical visualization, which LDA-Km can not provide.

4.5.2 Subspace Clustering

Subspace clustering is a challenge due to the curse of dimensionality. Meaningful clusters only exist in subspaces of the original feature space, for a survey see e.g. [72]. Basically, these methods can be categorized into two classes: axis-parallel subspace clustering and arbitrarily-oriented subspace clustering. The formal one try to find clusters in a subspace that formed by the subset of original features, e.g. PROCLUS [7], PreDeCon [22] and CLIQUE [10]. In this case, the clustering results are easy to interpret, but they cannot handle datasets with correlated features. The later one, arbitrarily-oriented subspace techniques, e.g. ORCLUS [9], 4C [23] and COPAC [4], are more related to our work. These methods assume each cluster being located in an unique arbitrarily-oriented subspace. The results are hard to interpret, since the detected subspaces are arbitrarily-oriented and there is no relationship among the clusters. Our aim is to find multiple low-dimensional subspaces that hierarchical arbitrarily-oriented subspace clusters can be best distinguished in the corresponding subspaces, which is different compared with existing methods. Subspace correlation clustering SSCC [56] try to analyze subspace projections to find subsets of objects showing linear correlations among this subset of dimensions. Subspace correlation analysis, e.g. REDUS [121] and CARE [120] is marginally related to this work. They try to identify subsets of original features that a group of data instances share a correlation. Therefore they find a unique subspace for each cluster as well. SSCC provides another way for interpretation on arbitrarily-oriented subspace clusters, however each detected cluster still own on unique subspace and there is no possibility to visualize the results.

4.5.3 Hierarchical Clustering

Hierarchical clustering algorithms compute a hierarchical decomposition of the data objects. The most widespread approaches to hierarchical clustering would be SingleLink [101] and OPTICS [14]. However these methods are not applicable when clusters of high-dimensional data only exist in subspaces. There are only a few works on hierarchical subspace clustering. HiSC [1] and DiSH [2] are proposed to find hierarchies of clusters ex-

isting in axis-parallel subspaces, while HiCO [5] and ERiC [3] aim at detecting hierarchies of arbitrarily-oriented subspace clusters. HiCO and ERiC are more related to our work. However, first they are designed to detect arbitrarily-oriented subspace clusters of different dimensionality which are nested into each other, which is different from our problem. Further, HiCO and ERiC find a unique subspace for each found cluster and it is not possible to interpret the results by visualization, which is one of the main contribution of MSS. Therefore, the proposed method MSS is different from the existing methods.

4.6 Conclusion

In this chapter, we introduce MSS, an effective and efficient clustering framework for detecting multiple low-dimensional subspaces, each exhibiting an interesting cluster structure. Besides, a hierarchy of clusters in multiple subspaces is provided for better interpretation the relationship among clusters. Integrating orthogonal Linear Discriminant Analysis, K-means and Kernel Density Estimation techniques, MSS can not only detect accurate clustering results, but also interpret them in multiple subspaces by hierarchical visualization. Besides, MSS is robust to irrelevant dimensions. Our extensive experiments on synthetic and real world data demonstrate that it is very useful to allow for multiple subspaces in clustering.

Chapter 5

Summarization-Compression Miner

Relational data now arises in various fields like transactional data, user-rating data, etc. In order to reveal the underlying patterns, an important task is co-clustering that partitions both types of data into meaningful clusters. Traditional co-clustering methods, as discussed in Chapter 2, usually only focus on the clustering part and ignore the relation between row and column clusters.

How to extract the truly relevant information from a large relational data set? The answer of this chapter is a technique integrating graph summarization, graph clustering, link prediction and the discovery of the hidden structure on the basis of data compression. Parts of the material presented in this chapter have been published in [45], where Xiao He and Jing Feng both contributed for the development of the main concept, Xiao He implemented the main algorithms and Jing Feng wrote the largest parts of the paper; Claudia Plant supervised the project and made contributions to the building of coding scheme; Bettina Konte performed part of experiments; Christian Böhm revised the whole paper; The co-authors also contributed to the conceptual development and paper writing.

> *"Jing Feng, Xiao He, Bettina Konte, Christian Böhm and Claudia Plant. Summarization-based Mining Bipartite Graphs. The 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2012: 1249-1257."*

Our novel algorithm SCMiner (for Summarization-Compression Miner) reduces a large

bipartite input graph to a highly compact representation which is very useful for different data mining tasks: 1) Clustering: The compact summary graph contains the truly relevant clusters of both types of nodes of a bipartite graph. 2) Link prediction: The compression scheme of SCMiner reveals suspicious edges which are probably erroneous as well as missing edges, i.e. pairs of nodes which should be connected by an edge. 3) Discovery of the hidden structure: Unlike traditional co-clustering methods, the result of SCMiner is not limited to row- and column-clusters. Besides the clusters, the summary graph also contains the essential relationships between both types of clusters and thus reveals the hidden structure of the data. Extensive experiments on synthetic and real data demonstrate that SCMiner outperforms state-of-the-art techniques for clustering and link prediction. Moreover, SCMiner discovers the hidden structure and reports it in an interpretable way to the user. Based on data compression, our technique does not rely on any input parameters which are difficult to estimate.

The remainder of this chapter is organized as follows: In Section 5.1, it starts with an introduction. We give the model formulation in Section 5.2. Section 5.3 presents the algorithm SCMiner in detail. Section 5.4 contains an extensive experimental evaluation. Section 5.5 briefly discusses related work and Section 5.6 concludes the chapter.

5.1 Introduction

Relational or graph-structured data are prevalent in nature, science and economics. Many aspects of real life can be well represented as a bipartite graph which has two types of vertices and edges representing the connections between them. Consider for example newsgroups where one type of vertices represents persons and the other type of vertices represent groups. An edge between a person and a group means that he or she is member of this group. Or consider the interaction between drugs and proteins: An edge between a particular substance and a protein means that the corresponding protein is responding to the drug. Such bipartite graphs are stored as large adjacency matrices often having millions of vertices and edges. Effective and efficient data mining methods are essential for an-

swering high-level domain specific questions like: Which users are potentially interested to join a newsgroup? Or, which combination of substances is most effective against a certain disease?

Therefore, in recent years the research topic of mining bipartite graphs has attracted much attention, with a large volume of research papers, e.g. [29, 37, 53, 77, 78, 90] to mention a few. To extract knowledge from a large bipartite graph, existing techniques apply one of the following two basic strategies: (1) Clustering. These approaches reduce the complexity by partitioning the large input graph into smaller groups of similar vertices which can be inspected and interpreted by the user. In particular, approaches to co-clustering like [29, 37] cluster both types of vertices, i.e. the rows and the columns of the adjacency matrix simultaneously guided by the idea that the clustering of rows and columns can profit from another. The output of co-clustering is a set of row-clusters and a set of column clusters. (2) Summarization. These approaches reduce the complexity not by grouping but by a global abstraction. The output of summarization techniques like [90] is a bipartite graph which is much smaller than the original input graph. Ideally, it distills the major characteristics from the input data and is small enough to be accessible for the user. A third line of chapter focuses on a different, however closely related topic: Link prediction. These approaches like [78] study the question whether there should be an edge between two currently disconnected vertices. Link prediction is closely related to summarization and clustering since a technique for link prediction must capture at least the local structure of the graph to provide reasonable predictions. Also clustering and summarization are closely related since during the process of abstraction, both approaches must identify the relevant major characteristics of the graph.

5.1.1 Contributions

In this chapter, we therefore propose SCMiner, a technique integrating summarization, clustering and link prediction on bipartite graphs. The name SCMiner stands for Summarization-Compression Miner and the principle of data compression also known as the

Minimum Description Length Principle (MDL) [97] is the basis of our technique. The basic idea is to transform the original graph into a very compact summary graph. During the process of transformation which is controlled by the MDL principle, our technique discovers the major clusters of both vertex types as well as the major connection patterns between those clusters. The result of SCMiner comprises a compressed graph, the row- and column-clusters and their link patterns. The major contributions of our approach can be summarized as follows:

1. **Clustering plus hidden structure mining.** Like state-of-the-art co-clustering methods, SCMiner accurately identifies the row- and column clusters of bipartite graphs and even outperforms them on some data sets. As a key feature of our approach, the result does not only consist of two sets of clusters. SCMiner also reveals the relationships between the clusters which are essential for interpretation.

2. **Accurate link prediction.** SCMiner accurately predicts missing or future links and removes noise edges.

3. **Unsupervised graph mining all in one.** SCMiner integrates summarization, clustering and link prediction and thus comprehensively supports unsupervised graph mining.

4. **Results validated by data compression.** Data compression is an intuitive optimization goal with many benefits: By natural balancing goodness of fit and model complexity, overfitting is avoided. The results are thus simple and interpretable. Moreover, supported by the MDL principle SCMiner does not rely on any difficult to estimate input parameters.

5.2 Compressing a Bipartite Graph

In this section, we first present a simple example to introduce the terminology of graph summarization. Then an MDL based coding schema is derived to find the best summa-

rization of a bipartite graph. At last, we propose a strategy to discover hidden relations between vertices of different type.

Notation and Example. Figure 5.1 depicts a simple example for graph summarization of a bipartite graph. $G = (V_1, V_2, E)$ is an unweighted bipartite graph where $V_1 = \{V_{11}, ..., V_{16}\}$ and $V_2 = \{V_{21}, ..., V_{26}\}$ denote two types of vertices and E denotes the edges between them. The summarization of graph G consists of two bipartite graphs, a summary graph G_S and an additional graph G_A. The summary graph $G_S = (S_1, S_2, E')$ is an aggregated graph and is composed of four super nodes $S_{11} = \{V_{11}, V_{12}, V_{13}\}$, $S_{12} = \{V_{14}, V_{15}, V_{16}\}$, $S_{21} = \{V_{21}, V_{22}, V_{23}\}$, $S_{22} = \{V_{24}, V_{25}, V_{26}\}$, and super edges E'. The super nodes themselves are composed of vertices exhibiting the exact same link pattern. They indicate the local cluster structure of each type of vertices, whereas the super edges represent the global relations between local clusters. The additional graph $G_A = (V_1, V_2, E'')$ contains normal nodes and correction edges which are required to reconstruct the bipartite graph G. The $+$ or $-$ symbols above the edges indicate whether it is required to add or remove them from G_S to recreate G. The correction edges describe the revealed hidden relations between different types of vertices.

5.2.1 Coding Scheme

A bipartite graph can be represented by thousands of summarizations, however, the challenge is to find out the one that best represents the data and reflects the hidden structure behind the surface data. The Minimum Description Length (MDL) principle is an intuitive choice for model selection [97]. It follows the assumption that the more we are able to compress the data, the more we have learned about its underlying patterns. Formally, the goodness of the model can be stated as shown in Eq.(5.1), where $L(M)$ denotes the cost for coding the model parameters and $L(D|M)$ represents the cost of describing the data D under the model M. As the model has to be coded with the data and too complex

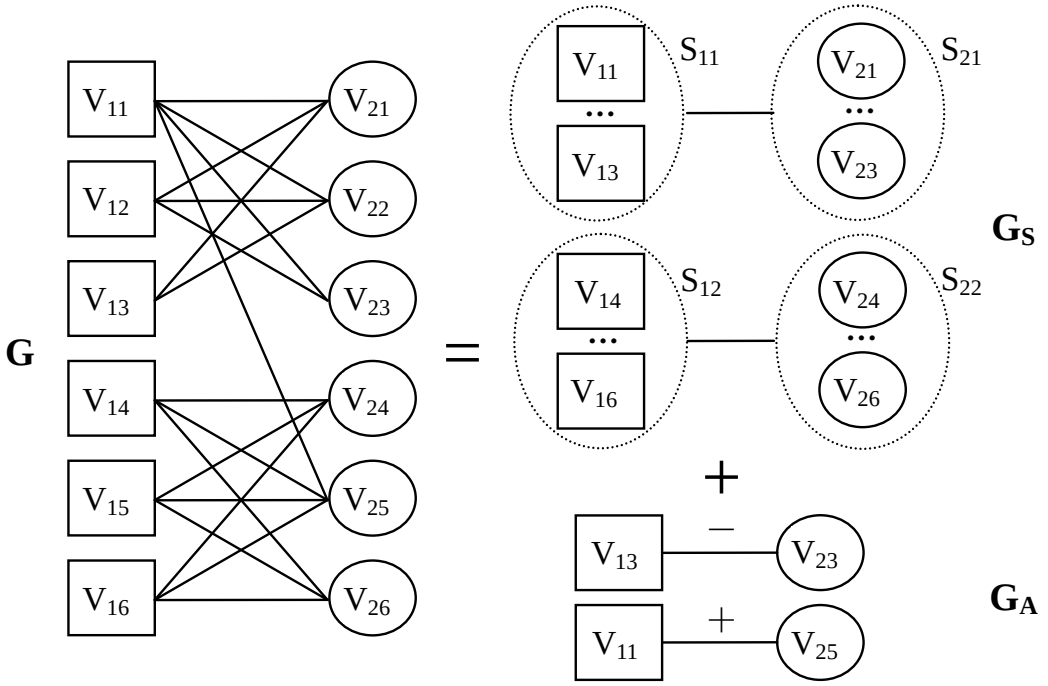


Figure 5.1: Summarization and compression of a bipartite graph.

models result in high compression cost, MDL naturally avoids overfitting.

$$L(M, D) = L(D|M) + L(M). \tag{5.1}$$

Inspired by the MDL principle, we propose a coding schema to choose the best graph summarization. The most direct and simple way to represent a bipartite graph $G = (V_1, V_2, E)$ is to compress its adjacency matrix $A \in |V_1| \times |V_2|$, with $a_{ij} = 1$ if $(V_{1i}, V_{2j}) \in E$. The coding cost of the adjacency matrix A is lower bounded by its entropy which is provided by:

$$CC(G) = -|V_1| \cdot |V_2| \cdot H(A), \tag{5.2}$$

where $H(A) = -(p_n(A) \cdot \log_2 p_n(A) + p_r(A) \cdot \log_2 p_r(A))$, $p_n(A)$ and $p_r(A)$ are the probabilities of finding 1 and 0 entries in the adjacency matrix A of G, i.e. the probabilities to observe edges or not.

As mentioned above, the summarization of a bipartite graph G is composed of two bipartite graphs, the summary graph G_S and the additional graph G_A. Instead of trans-

ferring G from a sender to a receiver, we can transfer G_S, G_A and the group information of super nodes in G_S. In addition, there is no need to send the symbol information of edges in G_A, since this information can be obtained by comparing G_A and G_S. If the symbol of an edge in G_A is $+$, this edge is not present in G_S, and vice versa, if the symbol is $-$, G_S contains the edge. The following equation defines the coding cost of a summarization of the graph.

Definition 5 (Coding Cost of a Graph.)**.**

$$CC(G) = CC(G_S) + CC(G_A) + \sum_{i=1}^{K} \sum_{j=1}^{N_i} |S_{ij}| log_2 \frac{|V_i|}{|S_{ij}|}, \tag{5.3}$$

where $CC(G_S)$ and $CC(G_A)$ denote the coding cost of summary graph G_S and the additional graph G_A defined by Eq.(5.2). The third term is the cost of coding the group information, where K is the number of node types, N_i is the number of super nodes in type i, $|S_{ij}|$ is the number of members in super node S_{ij}, $|V_i|$ is the number of original nodes in type i.

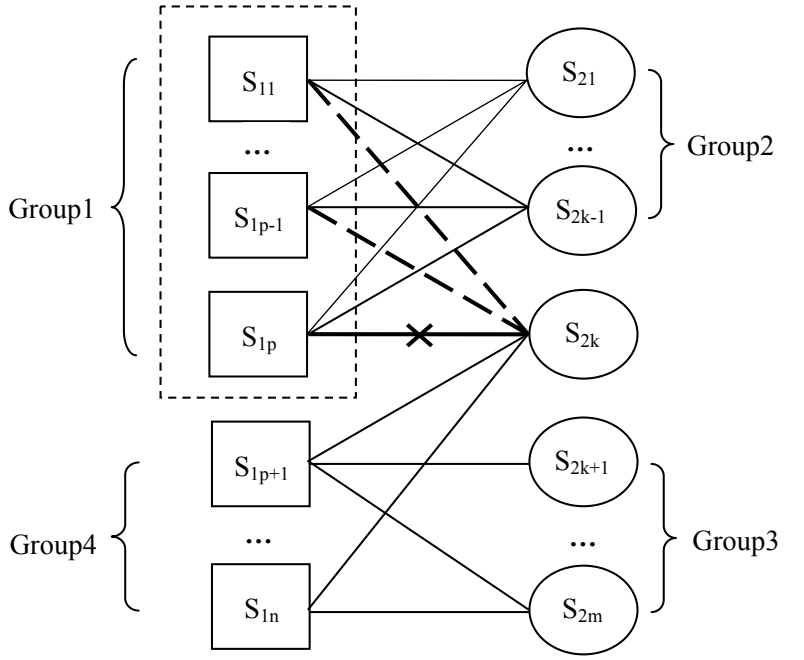
$L(D|M) = CC(G_S)$ is the description of the data under the model M with coding cost $L(M) = CC(G_A) + CC(group)$. The optimization goal of our algorithm SCMiner is to find the best model or summarization that minimizes Eq.(5.3).

5.2.2 Hidden Relations Between Vertices

The data we record in real life applications does often only approximately represent the ground truth due to inconsistencies and measurement errors. For example, the same user often joins newsgroups under different nicknames and email addresses. Or a protein might show a strong response to a substance simply due to a measurement error. Thus real world graphs are often spoiled by erroneous connections on the one hand and are also missing important links on the other hand.

Take a close look at the graph G in Figure 5.1, obviously we have the feeling that edge (V_{13}, V_{23}) is probably missing, and edge (V_{11}, V_{25}) maybe presents an artifact of noise. Therefore, if we add edge (V_{13}, V_{23}) to G and remove edge (V_{11}, V_{25}) from G, we can form

four super nodes, whose members exhibit the exact same connection patterns. These connections probably reflect the true relationships. Based on these observations, we propose the following strategy to discover the hidden relations between vertices.



Group1: Nodes to be merged
Group2: Hop 2 neighbors of S_{2k} and common neighbor of Group1
Group3: Hop 2 neighbors of S_{2k} but not common neighbor of Group1
Group4: Neighbors of S_{2k}

: The edge to be deleted
: The edge to be added

Figure 5.2: The strategy of SCMiner used for merging nodes

Figure 5.2 describes our strategy for merging nodes, suppose all the nodes are super nodes. The nodes in Group 1 share a similar link pattern and could therefore be merged into a super node. Nodes in Group 2 and Group 3 are both hop two neighbors of node S_{2k}, specifically nodes in Group 2 are common neighbors of nodes in Group 1 and nodes in Group 3 are not. To form a new super node of nodes in Group 1 their link pattern should be exactly the same. However, the link pattern of node S_{1p} in Group 1 is similar to those

Algorithm 5 ***ModifyEdge***

```
//Modify edges of group to make their link same
Input: Group nodes group, G_S, G_A
Output: G_S, G_A, hop2Sim
alln = Neighbor(group);
cn = CommonNeighbor(group);
for Each node S ∈ alln and S ∉ cn do
   Using Eq.(5.4) and Eq.(5.5) to add or remove edge;
   Add or remove edges in G_S;
   Add additional edges to G_A;
end for
Update hop2Sim for each S ∈ alln and S ∉ cn;
return G_S, G_A, hop2Sim;
```

of the other nodes, but not the same. Concretely, S_{1p} has an extra link with S_{2k} that the other nodes lack. As we can see from Figure 5.2, two methods can be adopted to make nodes in Group 1 exhibiting the same link pattern: we could either remove edge (S_{1p}, S_{2k}) or add edges $(S_{11}, S_{2k}), (S_{12}, S_{2k}), ..., (S_{1p-1}, S_{2k})$. The question is, how to distinguish whether edges should be removed or added?

We can decide if we add or remove edges by calculating some cost function. We define the cost as the number of edges we add or delete, which is highly related to the MDL costs of Eq.(5.3), in order to make the link patterns of merging nodes equal. Specifically, as shown in Figure 5.2, there are p nodes in Group 1 $S_{11}, S_{12}, ..., S_{1p}$, among which S_{1p} exhibits a similar but not exact same link pattern as the other nodes. Therefore, the cost of removing edge (S_{1p}, S_{2k}) is 1 and the cost of adding edges $(S_{11}, S_{2k}), (S_{12}, S_{2k}), ..., (S_{1p-1}, S_{2k})$ is $p-1$. Obviously, the former method should be selected because of the lower cost. The pseudocode of modifying the edges of a merging group is shown in Algorithm 5. At first we combine neighbors and common neighbors of nodes in merging groups, then we need to modify the edges of those nodes which are neighbors but not common neighbors of the merging group to make the link pattern of all group members exactly the same. Suppose the merging group contains $S_{11}, S_{12}, ..., S_{1p}$ and S_{2k} is one of their neighbors but not a common neighbor. The cost of removing and adding edges can be calculated by Eq.(5.4) and Eq.(5.5). The

cost of a super edge is calculated as $|S_{1i}| \cdot |S_{2k}|$, where $|.|$ is the number of normal nodes contained in a super node. However, in some cases the costs for removing might be the same as for adding edges. If so, we use the similarity proposed in the next section to decide which action to take. Naturally, if S_{2k} is more similar to the nodes of Group 2 compared to those of Group 3, we add edges for merging, while otherwise we remove edges.

$$Cost_{remove} = \sum_{i=1}^{p} \begin{cases} |S_{1i}| \cdot |S_{2k}| & \text{if } S_{1i} \text{ links to } S_{2k} \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

$$Cost_{add} = \sum_{i=1}^{p} \begin{cases} 0 & \text{if } S_{1i} \text{ links to } S_{2k} \\ |S_{1i}| \cdot |S_{2k}| & \text{otherwise.} \end{cases} \tag{5.5}$$

5.3 Algorithm SCMiner

In this section, we propose our algorithm SCMiner to find the best summarization of a bipartite graph. It can be proven that finding the global optimal summarization is NP-hard, therefore SCMiner follows a heuristic approach to search the local optima.

Basic Idea. The idea behind the algorithm SCMiner is that nodes with similar link patterns can be merged to groups, if the similarity between each pair of nodes in the group is bigger than some threshold th. By adding and removing edges as proposed in Section 2 we accomplish that all group nodes share the exact same link pattern and so can form a super node. SCMiner iteratively merges groups of nodes or super nodes whose similarities are bigger than th. The initial threshold is 1 and th is reduced stepwise by ϵ when no pair of nodes can be merged. In each iteration we calculate the new summarization coding cost. Then the MDL principle is used to choose the best summarization. The algorithm terminates when th reaches 0.

Finding Super Node Candidates. To find suitable candidate groups of nodes where merging might pay-off, we introduce the notion of hop two similarity. Suppose two super nodes S_{1i} and S_{1j} have n common neighbors $\{S_{21}...S_{2n}\}$ and m neighbors $\{S_{2(n+1)}...S_{2(n+m)}\}$ that are connected to only one of the two super nodes. Intuitively, if the two nodes have

more common neighbors than neighbors only linking to one node, they are more similar in link pattern, then different. We can define their similarity as

Definition 6 (Hop Two Similarity.)**.**

$$sim(S_{1i}, S_{1j}) = \frac{\sum_{k=1}^{n} |S_{2k}|}{\sum_{k=1}^{n+m} |S_{2k}|} \tag{5.6}$$

where $|S_{2i}|$ is the number of normal nodes contained in super node S_{2i}. This similarity measure ranges from 0 to 1.

As described above, we only need to calculate the similarity between two nodes if they share at least one common neighbor and therefore are hop two neighbors of each other. In the following we call the similarity between a node and all its hop two neighbors its hop two similarity.

Algorithm. Now we describe our algorithm SCMiner, the pseudocode is provided in algorithm 6. The input parameters of SCMiner are the bipartite graph $G = (V, E)$, where $V = (V_1, V_2)$ and E are the edges between V_1 and V_2, and stepsize ϵ for reducing th. The output of SCMiner is the summarization of G, including the summary graph $G_S = (S, E_S)$ composed of super nodes $S = (S_1, S_2)$ and the additional graph $G_A = (V, E')$ composed of single nodes $V = (V_1, V_2)$, which has the minimum coding cost regarding the proposed coding scheme. In the initialization phase, we first set the summary graph G_S to the input graph G, which means that all single nodes are treated as super nodes containing only one normal node, and set the additional graph G_A empty. Then we initialize the coding cost *mincc* using *Eq.*(5.3) with G_S and G_A. Afterwards we compute the similarities between each node and its hop two neighbors of same node type. In the searching phase, when $th > 0$, we do the following steps: First we search for groups of nodes that have at least one hop two neighbor with similarity larger than th and then we merge every group we found. When there is no more group of nodes that can be merged, we decrease the threshold th by ϵ. In the merging phase, we use the proposed method shown in algorithm 5 to modify the edges of the merging group that are present in G_S to get nodes with exact same link

Algorithm 6 ***SCMiner***

Input: Bipartite graph $G = (V, E)$, Reduce step ϵ
Output: Summary Graph G_S, Addition Graph G_A

//Initialization
$G_S = G$, $G_A = (V, \emptyset)$;
Compute $mincc$ using $Eq.(5.3)$ with G_S and G_A;
$bestG_S = G_S$, $bestG_A = G_A$;
Compute $hop2Sim$ for each $S \in G_S$ using $Eq.(5.6)$;

//Searching for best Summarization
while $th > 0$ **do**
  for each node $S \in G_S$ **do**
    Get SN with $S' \in SN$ and $hop2Sim(S, S') > th$;
  end for
  Combine SN and get non-overlapped groups $allgroup$;
  for each $group \in allgroup$ **do**
    ModifyEdge$(group, G_S, G_A)$;
    Merge nodes of G_S with same link pattern;
    Compute cc using $Eq.(5.3)$ with G_S and G_A;
    Record $bestG_S$, $bestG_A$, and $mincc$ if $cc < mincc$;
  end for
  if $allgroup == \emptyset$ **then**
    $th = th - \epsilon$;
  else
    $th = 1.0$;
  end if
end while
return $bestG_S$, $bestG_A$;

structure. Subsequently we add the corresponding additional edges to G_A and update the hop two similarity for affected nodes, which are neighbors of merging group nodes but not their common neighbors. During the merging phase, nodes with similar link patterns are merged, which decreases the data cost and increases the model cost, while the total cost is reduced in most cases. After each merging step, we calculate the coding cost using Eq.(5.3) with current G_S and G_A and the best summarization with minimum coding cost is stored in $bestG_S$ and $bestG_A$. Finally, we output $bestG_S$ and $bestG_A$ with coding cost.

Properties. We adjust the parameter ϵ using the MDL principle. Extensive experiments on synthetic and real data showed that a suitable range for ϵ is around 0.01 to 0.1.

Therefore we try out every setting with a step size of 0.01 and let MDL select the best result. The runtime complexity for the computation of the similarity between each vertex v and its hop two neighbors is $O(N \cdot d_{av}^3)$, where N is the number of vertices and d_{av} is the average degree of each vertex. During each merging step in SCMiner, we only compute the similarities between some affected vertices and their two hop neighbors. Therefore the runtime complexity is roughly $O(d_{av}^4)$. The number of merging steps, affected by the reduction stepsize ϵ, is N in average. So the whole runtime complexity is $O(N \cdot d_{av}^4)$.

5.4 Experiments

This section provides empirical evidence to show the effectiveness of SCMiner on synthetic and real data. In particular, we evaluate SCMiner three aspects: First, we compare SCMiner with state-of-the-art co-clustering algorithms in terms of quality of the clusters detected on each type of nodes. Secondly, we evaluate the hidden structure, i.e. the relationships between both types of nodes found by SCMiner. Finally, we compare SCMiner with some state-of-the-art link prediction methods to evaluate the validity of hidden relationships discovered by SCMiner.

Data Sets. The generated synthetic bipartite graphs with different parameters are shown in Table 5.1. Both V_1 and V_2 contain the same number of clusters, and each cluster has 100 nodes. The matrix T shows the ground truth links between clusters of different type, where T_{pq} can take the values 1 or 0, which means the pth cluster of V_1 and the qth cluster of V_2 are fully connected or separated. The matrix S introduces link parameters to matrix T, where S_{pq} denotes the percentage of links generated between the pth cluster of V_1 and the qth cluster of V_2. In other words, if link parameters are introduced based on 0, noisy links are added to the ground truth graph. If link parameters are introduced based on 1, links are removed from the ground truth graph, where the percentage is 1 minus the link parameters.

Three real data sets are evaluated in our experiments. World cities[1] data consists of

[1] http://www.lboro.ac.uk/gawc/datasets/da6.html

Table 5.1: Synthetic bipartite graphs

Data Set	S	T
BP1	$\begin{pmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
BP2	$\begin{pmatrix} 0.9 & 0.8 & 0.1 \\ 0.1 & 0.9 & 0.8 \\ 0.1 & 0.2 & 0.9 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
BP3	$\begin{pmatrix} 0.8 & 0.7 & 0.2 & 0.8 \\ 0.9 & 0.3 & 0.8 & 0.2 \\ 0.3 & 0.8 & 0.2 & 0.7 \\ 0.9 & 0.8 & 0.7 & 0.2 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

the distribution of offices from 46 global advanced producer service firms over 55 world cities. Global firms are defined as firms owning offices in at least 15 different cities. Service values for a firm in a city are given as 3,2,1 and 0. We binarize the data set such that positive service values become 1 and then generate the bipartite graph. The advanced producer service firms can be categorized into 4 clusters: accountancy firms, advertising firms, banking and finance firms and law firms.

MovieLens[2] data was collected through the MovieLens web site during the seven-month period from September 19th, 1997 to April 22nd, 1998 by the GroupLens Research Project at the University of Minnesota. It consists of 100,000 ratings (1-5) from 943 users on 1682 movies. We preprocess the data by removing movies which are rated by less than 30 users and users that rated less than 30 movies. Therefore, we got a bipartite graph of 361 users and 334 movies. Then we binarize the preprocessed data set such that 3-5 entries become 1 and others become 0.

Jester[3] is a joke rating data set. The original data set contains over 1.7 million continuous ratings (-10.00 to +10.00, +10.00 best) of 150 jokes from 63974 users collected between April 2006 to May 2009. We remove 22 jokes which are never rated or rated by less than 0.01 of users, and randomly pick 1000 users who rate all the picked jokes. Then we binarize the data set such that 5-10 entries become 1 and others become 0. The ground

[2]http://www.grouplens.org/node/12
[3]http://eigentaste.berkeley.edu/dataset/

truth is generated for evaluating link prediction, such that the non-negative entries become 1 and the negative entries become 0.

5.4.1 Clustering Quality

Firstly, we evaluate the quality of clusters detected by SCMiner. SCMiner can be used as a parameter-free bipartite graph partition method and therefore we choose the approaches Cross-association (CA) [29] and Information-theoretic Co-clustering (ITCC) [38] as comparison methods. In addition, we compare SCMiner to the graph summarization technique (GS) of [90]. The algorithms SCMiner, CA and GS are both parameter-free, ITCC requires the number of clusters in rows and columns. The algorithm GS does not output a clustering, however, the summarized nodes can also be regarded as clusters. We therefore create a cluster for each summarized node of the algorithm. For synthetic data we set the number of clusters of ITCC to the true number of the data set. We use the Normalized Mutual Information (NMI) [107] to measure the clustering performance. The value of NMI ranges between 0 and 1. The higher the value the better the clustering. We further report the Adjusted Mutual Information (AMI) and Adjusted Variation Information (AVI) scores proposed in [107].

Table 5.2: Clustering Performance on Synthetic data.

	BP1	BP2	BP3
SCMiner	**1**	**1**	**0.9949**
CA	0.6683	0.7897	0.8750
ITCC	**1**	**1**	0.8750
GS	0.2568	0.4069	0.5493

Synthetic Data. Table 5.2 depicts the clustering performance comparison on synthetic data sets evaluated by NMI, which shows that SCMiner yields better results than CA, ITCC and GS. For BP1 and BP2 both ITCC and SCMiner give perfect results, however ITCC needs the true number of clusters as input parameter, whereas SCMiner determines the number of clusters automatically without user input. CA outputs worse NMI results,

because it splits the clusters into some smaller ones. Worst NMI results yields GS, this is mainly because GS is designed for summarization but not for clustering. For space limitations, Table 5.2 reports only NMI but the other scores are similar.

For SCMiner we try out several ϵ and choose the best results with the minimum MDL. Figure 5.3 shows the clustering results for all synthetic data sets with different ϵ and one can see that the results are quite stable on a wide range of ϵ. We also test the relationship between NMI and MDL, for space limitation we only mark several MDL values for synthetic data BP3 in Figure 5.3. The coding costs are 127102 bits when $\epsilon = 0.01$ and $NMI = 0.89$, and it costs 123723 bits when $\epsilon = 0.07$ and $NMI = 0.99$, which proofs that smaller MDL values lead to better NMI results.

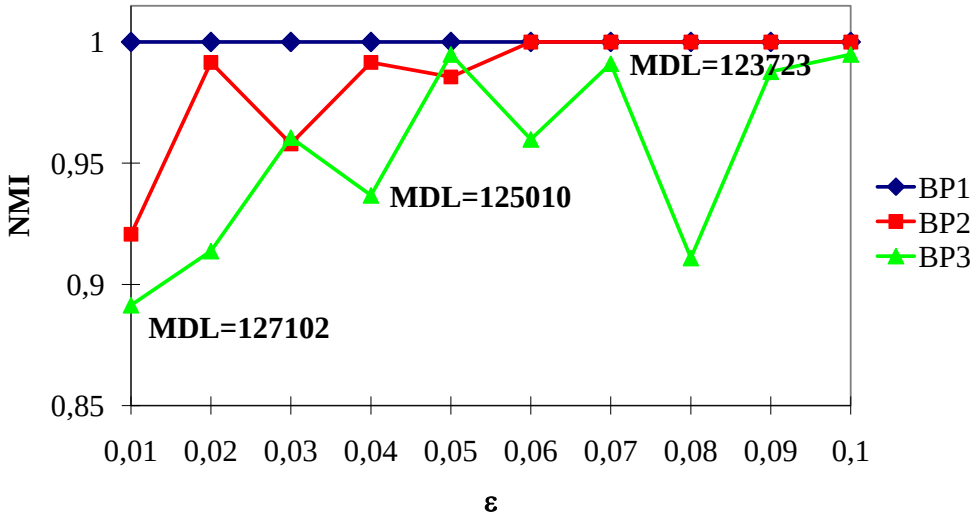


Figure 5.3: Results of SCMiner on synthetic data sets for various ϵ.

World Cities Data. The World cities real data set contains cluster labels of global firms and thus we can use NMI to evaluate clustering quality of this type of nodes. We run SCMiner, CA, ITCC and GS on this data set. SCMiner, CA and GS are all parameter-free methods (the chosen ϵ of SCMiner for cities is 0.01). We set the true number of clusters of global firms and 4 as the number of clusters of world cities as input parameter for ITCC. The NMI results for global firms are depicted in Table 5.3 and the table shows that SCMiner clearly outperforms CA, ITCC and GS in all clustering quality scores. The evaluation of cities type clusters is much more difficult, since the cluster labels are not provided along

Table 5.3: Results on World Cities Data.

	NMI	AMI	AVI
SCMiner	**0.4345**	**0.3807**	**0.3824**
CA	0.3109	0.2447	0.2604
ITCC	0.2522	0.1845	0.1891
GS	0.2515	0.0467	0.0683

with the data set. Looking in detail at the cluster contents, SCMiner finds 3 clusters and a separated city Washington, DC. The first cluster contains 13 cities, including Atlanta, Boston, Dallas, Munich, Montreal etc. and all these cities are strong in economics. In detail, all 5 accountancy firms have services in these cities, and there are 3 advertising, 5 banking and 2 law companies in average owning offices in these 13 cities. The second cluster contains 17 cities, including Toronto, Paris, London, New York, and Beijing. All these cities are metropolis in the world. Moreover most of these cities are capital of their country. In terms of service, all 5 accountancy firms have services in these cities, and there are 9 advertising, 11 banking and 7 law companies in average having offices in these 17 cities. The third cluster contains 24 cities, including Amsterdam, Barcelona, Seoul, Shanghai etc., which are all kind of financial cities. In terms of service, all 5 accountancy firms have services in these cities as well, and there are 7 advertising, 8 banking and 2 law companies in average having offices in these 24 cities. The cluster analysis shows that SCMiner outputs reasonable clusters regarding the cities type nodes. To sum up, the cities can be categorized into 3 types, capital metropolis, financial cities and economic cities. Capital metropolis are the economic, financial and politic center of a country, therefore all kinds of companies have offices in these cities. Financial cities offer a lot of banking and advertising company services, but lack law services, because they are not politically oriented. Some local manufactories are located in economic cities, whereas advertising, banking and law companies are not. Washington, DC is quite different compared to the other cities, since it is a politic, but not a financial city and therefore owns lots of law firms' offices but fewer advertising and banking firms' offices. Thus it is reasonable that this city is separated in its own cluster.

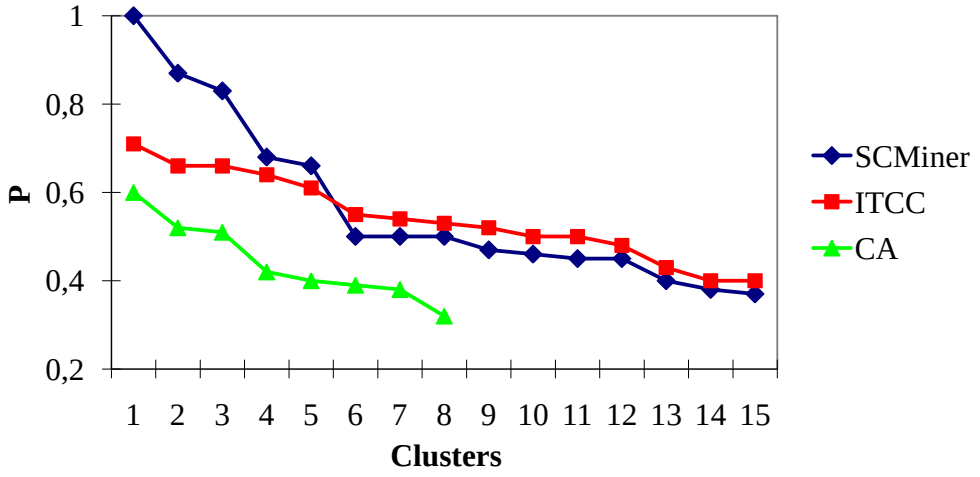


Figure 5.4: Cluster purity comparison on Movielens Data.

Movielens Data. MovieLens data set does not provide cluster labels, however, it provides movie categories and personal information of users that can be used to analyze the cluster contents. We separately perform SCMiner, CA, ITCC and GS on this data set, SCMiner, CA and GS are both parameter-free (the chosen ϵ of SCMiner for MovieLens is 0.05). CA outputs 9 user clusters and 8 movie clusters. In addition some isolated nodes and small clusters with less than 5 instances are found. SCMiner gives 10 user clusters and 15 movie clusters, whereas all the clusters found by GS are single nodes or just contain a few nodes (< 5). To be fair, we set the number of user and movie clusters to 10 and 15 for ITCC, which corresponds to the number of clusters found by SCMiner. In this data set, movies are classified into 19 genres, such as action, adventure, animation, etc, and a movie can be categorized into several genres at once. According to the basic concept of clustering that objects in the same cluster are similar, we define purity P to evaluate the movie clusters. By counting genres of movies, we can acquire the genre that dominates the cluster and let this genre be the cluster representative. The purity of a cluster is defined as the percentage of movies belonging to the most represented genre. We calculate the purity P for each movie cluster detected by SCMiner, CA and ITCC, and sort them by p value, which is shown in Figure 5.4. The figure shows that SCMiner outputs more pure movie clusters than the two comparison methods. In terms of user clusters, it is difficult to

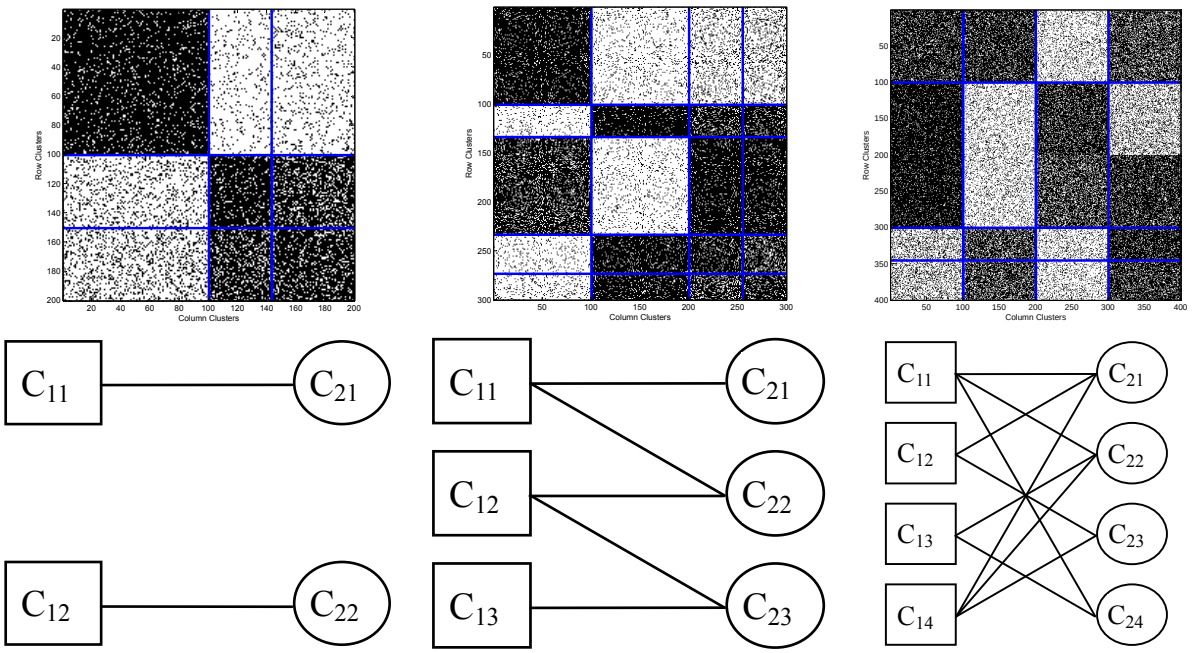


Figure 5.5: Hidden Structure Detected by SCMiner and CA. From Left to Right: Data Set BP1, BP2, BP3.

compare the results. Analyzing the content of user clusters detected by SCMiner, reveals that some clusters contain old users, some are composed of young users, some groups contain only males, while others own mainly women. This reflects that SCMiner outputs meaningful user clusters.

5.4.2 Hidden Structure

For bipartite graph data, we do only want to analyze the clusters in each type, but we also want to evaluate the relationship between clusters of different types. Besides our SCMiner algorithm, CA is the only comparison method providing information about these relationships (in the form of a re-arranged adjacency matrix) to users.

Synthetic Data. Figure 5.5 depicts the hidden structure detected by SCMiner and CA, the top row shows the re-arranged adjacency matrix of data output by CA, the bottom row depicts the hidden structure found by SCMiner, where squares and circles denote the two different types of clusters. For BP1 and BP2 SCMiner gives perfect results and for BP3

just one node is categorized incorrectly, as cluster C_{12} contains 99 nodes and clusterC_{14} contains 101 nodes, but the whole structure is correct, whereas the cross association output by CA for BP3 is incorrect and it is really hard to tell the relationship between the second row and fourth column cluster. In summary, the visualization of cross-associations is only clear and interpretable for data having a relatively easy structure.

World Cities Data. Figure 5.6 depicts the hidden structure of World cities data set detected by SCMiner and CA. The left shows the re-arranged adjacency matrix of data output by CA. It is hard to identify the relationship between two types of nodes from the matrix itself. The right depicts the hidden structure found by SCMiner, the squares represent the cities clusters and circles denote clusters of companies. The figure shows that capital metropolis have connections to all kind of firms, financial cities do not have law companies services, most service in economic cities are from accountancy companies, and Washington, DC owns lots of law companies but fewer banking and advertising firms. From the figure, the major structure of the complex network becomes obvious at first glance. The result of SCMiner is a highly compact bipartite graph, a data representation which is easy to understand for the user. In contrast, the re-arranged adjacency matrix is still quite noisy and it is very difficult to infer the structure from this representation.

Movielens Data. Figure 5.7 depicts the hidden structure of MovieLens data set detected by SCMiner and CA, the left shows the re-arranged adjacency matrix of data output by CA, which is hard to understand, the right depicts the hidden structure found by SCMiner which is much easier to analyze. For clarity we only show the relation between the 5 biggest clusters in each type. Squares and circles represent the user and movies cluster. Cluster $O1$ and $O2$ contain user groups that are older than the average regarding the whole data set, and their male-to-female ratio is just about whole data set average. Square YW represents a user group with a much larger female ratio compared to the average, and the users are younger than the average as well. Square $YM1$ and $YM2$ denote clusters with groups of users containing almost all young man. On the other side, circle FM represents a cluster of famous movies, containing Schindler's List, Shawshank Redemption, and Seven etc. that all have high rating scores in IMDB, which shows that

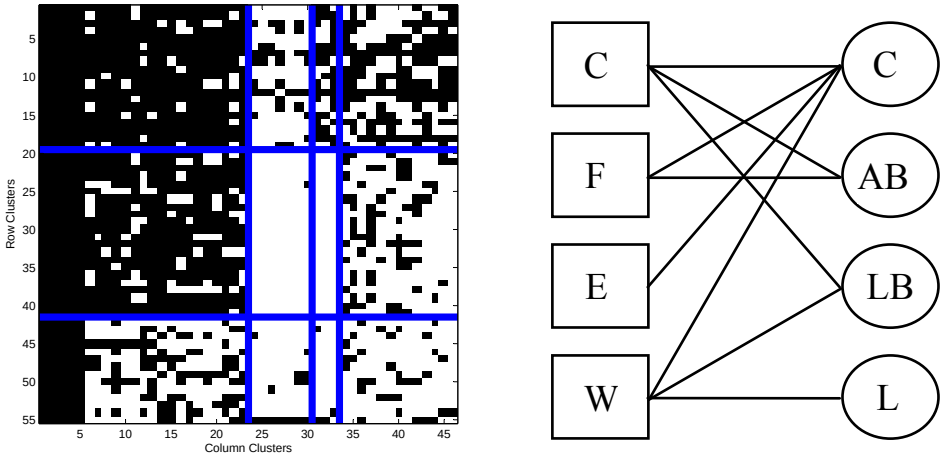


Figure 5.6: Hidden structure of cities detected by CA (left) and SCMiner (right). For SCMiner results, Square C represents a cluster consisting of capitals, Square F financial cities, Square E economic cities and Square W is the isolated city Washington, DC. On the other side, the cluster represented by Circle C mainly contains accountancy firms, Circle AB advertising and banking companies, Circle LB banking and law firms, and Circle L law firms.

all groups of users like it. This fact is embodied in the revealed hidden structure, since this cluster has connections to all user groups. Circle CD is mainly consisting of movies from comedy and drama category, so it makes sense that older people like it. The movies in circle CR are mostly from action, romance and comedy genre, so young women and young men like it. Circle AA and AS represent adventure action thriller movies and action sci-fi thriller movies respectively, therefore young man like them. The above analysis shows that the hidden structure found by SCMiner is reasonable.

5.4.3 Link Prediction Accuracy

SCMiner outputs the summarization of the input graph that yields the minimum coding cost. The summarization itself consists of a summary graph and an additional graph, where the "-" edges in the additional graph denote missing links which should be added to the original graph, and "+" edges might be noisy links which should be removed from the original graph. Since it is hard to determine whether a "-" edge represents noise, we only use link prediction to evaluate whether a "+" edge is a missing link.

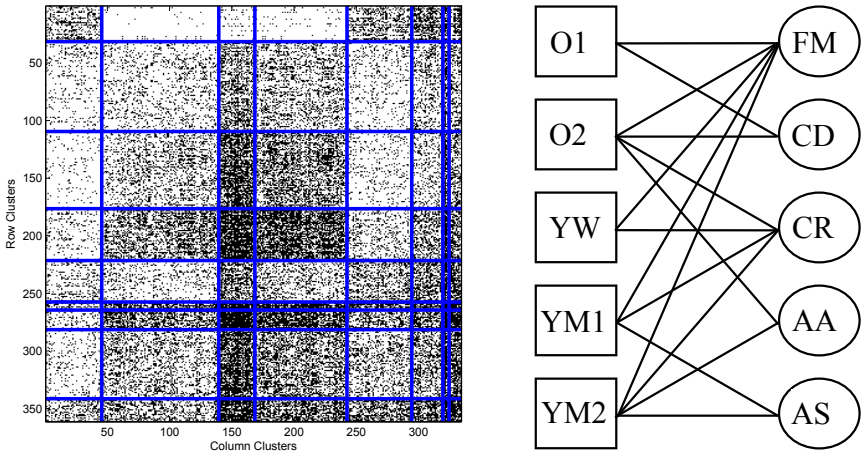


Figure 5.7: Hidden structure of MovieLens detected by CA (left) and SCMiner (right). For SCMiner results, Square $O1$ and $O2$ denote clusters containing old users, Square YW represents a cluster of young women, and Square $YM1$ and $YM2$ young man. On the other side, the cluster represented by Circle FM represents high scored movies, Circle CD comedy and drama, Circle CR action, romance and comedy. Circles AA and AS represent adventure, action, thriller movies and action, sci-fi, thriller movies, respectively.

The link prediction problem has been studied on graphs by several approaches which can be divided into two categories, supervised, e.g. [78], [59] and unsupervised methods, e.g. [77]. Since SCMiner is unsupervised, we only compare our results to unsupervised approaches. There are two types of unsupervised methods: based on node neighbors and based on paths between nodes. However, some of these methods, like common neighbor, are not suitable for link prediction on bipartite graph data and therefore we choose two methods, *PA* preferential attachment [91] and *Katz* [68], that are suitable for bipartite graphs. We use precision and recall to evaluate the accuracy. SCMiner automatically outputs K predicted edges and for reasons of fairness we choose the top K predictions of the ranking list of *PA* and *Katz* to compare the precision and recall of the prediction results. We also compare our approach to GS Graph Summarization, since the algorithm changes the edges of the original graph as well. Table 5.4 shows the link prediction comparison on our synthetic data sets as well as on the real data set Jester. All the results are averaged over runs on 10 randomly generated or the selected data sets. SCMiner predicts the missing links of BP1 and BP2 completely and those of BP3 nearly completely correct and therefore

Table 5.4: Link Prediction Performances.

Algorithm	BP1		BP2		BP3		Jester	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
PA	0.084	0.084	0.436	0.436	0.442	0.443	0.406	0.369
Katz	0.984	0.984	0.943	0.943	0.547	0.548	0.406	0.369
GS	0.965	**1**	0.981	1	0.994	0.973	0.356	0.05
SCMiner	**1**	**1**	**1**	**1**	**0.997**	**1**	**0.431**	**0.389**

yields better results on all synthetic data sets than all comparison methods. Moreover it outperforms the other approaches on the Jester data set. Interestingly, GS performs better on the synthetic data sets than on Jester. The reason might be that the Jester data set is sparser than the synthetic data sets, and therefore GS reaches a local minimum very fast, which results in fewer predicted edges.

5.5 Related Work

During past decades, many algorithms were proposed for graph clustering, graph compression, graph summarization and link prediction. Due to space limitations, we only provide a very brief survey on some related research directions.

5.5.1 Co-clustering

Bi-clustering or co-clustering is a creative approach which simultaneously clusters rows and columns of a data matrix. It avoids the problems caused by sparseness and high-dimensionality that traditional one way clustering algorithms suffer from. There are some state-of-the-art co-clustering algorithms, such as Information- theoretic Co-clustering [38], Cross Association [29] etc. Specifically, Information theoretic co-clustering [38] simultaneously maps row elements to row clusters and column elements to column clusters, mutual information of each clustering state is calculated and compared to the initial state. Thus, the optimal co-clustering result is obtained when the mutual information loss is minimal. However, the drawback of the method is that the number of both row and column

clusters must be predetermined. Cross Association [29] is a MDL-based parameter-free co-clustering method that processes a binary matrix and seeks clusters of rows and columns alternately. Then the matrix is divided into homogeneous rectangles which represent underlying structure of the data. However, compared with our algorithm, it only addresses the clustering problem. Moreover, Long et al. [80] proposed a framework for co-clustering named block value decomposition (BVD), which formulates co-clustering as an optimization problem of matrix decomposition. Similarly, in [79], Long et al. reconstruct a bipartite graph based on some hidden nodes and thus an optimal co-clustering result is obtained from the new bipartite graph which mostly approximates the origin graph. However, these two algorithms both need the number of clusters as input parameter. Besides, Dhillon [37] proposed a spectral algorithm for bipartite graph partition. Shan and Banerjee [98] proposed a Bayesian co-clustering model and considered co-clustering as a generative mixture modeling problem. George and Merugu proposed a co-clustering algorithm based on the collaborative filtering framework [53]. In terms of real life data sets, [32] applied co-clustering on gene expression data and [37] effectively processed documents and words data.

5.5.2 Graph Compression and Summarization

In real life, it is common that a graph consists of tens of thousands nodes and complex linkages. Graph compression and graph summarization [90, 106, 117] are effective ways to simplify the original large graph and to extract useful information. Many algorithms are based on Subdue [64], which is a classical graph compression system. It makes use of the MDL Principle to find a subgraph suitable for compression. Bayesian Model Merging (BMM) [103, 104] also is a kind of compression method which is based on Bayesian formula. The best compressed model can be achieved by maximizing the posterior probability. Navlakha et al. [90] represented a graph by a summary graph and a correction matrix, and summarization is implemented by greedy merging and randomized algorithms. MDL is used to find the local optimum. However, the algorithm is designed for compressing only

and performs poorly in clustering or link prediction. In [106], according to group nodes based on their attributes and relationships, which are selected by the user, Tian et al. propose a graph summarization algorithm.

5.5.3 Link Prediction

Predicting whether two disconnected nodes will be connected in the future, based on available network information is known as the challenge of link prediction. Liben-Nowell and Kleinberg [77] summarize some unsupervised link prediction approaches for social networks. For example, common neighbor, Jaccard's coefficient, preferential attachment [91], *Katz* [68] etc. Specifically, preferential attachment is based on the idea that two nodes with higher degree have higher probability to be connected. *Katz* [68] defines a score that sums up the number of all paths between two nodes where short paths are weighted stronger. By ranking these scores, the probability of whether two disconnected nodes will be linked in the future can be acquired. However, when dealing with bipartite graphs, Kunegis et al. [75] stated that scores based on common neighbor are not suitable, because two connected nodes do not have any common neighbors in a bipartite graph. But scores like preferential attachment and *Katz*, which are used in this chapter as comparison methods are reasonable.

5.6 Conclusion

In this chapter, we propose SCMiner, a technique that integrates graph summarization, bipartite graph clustering, link prediction and hidden structure mining. Based on the sound optimization goal of data compression, our algorithm finds a compact summary representation of a large graph. In addition, while compressing the graph SCMiner detects the truly relevant clusters of both node types and their hidden relationships. Thus, SCMiner is a framework comprehensively supporting the major tasks in unsupervised graph mining. Our experiments have demonstrated that SCMiner outperforms specialized state-of-the-art

techniques for co-clustering and link prediction. In ongoing and future work we want to extend this idea to support multi-partite graphs as well as graphs with different edge types.

Chapter 6

Probabilistic Integral Metric for Multi-instance Data

In this chapter, we study another type of complex high-dimensional data: Multi-instance data. In many application domains, like life science, multimedia retrieval, etc., data is often modeled as multi-instance objects. How to define a useful similarity measure for multi-instance objects? This topic has received considerable attention during the last years with approaches like the Hausdorff distance (HD), the Sum of Minimum Distances (SMD), Relational Instance Based Learning (RIBL) etc. However, it remains unclear what are precisely the intuition, the basic assumptions and finally, which of these similarity measures is suitable for a given application scenario.

To overcome this lack of substantiation and to bridge the gap between similarity measure and application, in this chapter we define a generative probabilistic model for the multi-instance objects suggesting a novel similarity measure: The Probabilistic Integral Metric (PIM). Parts of the material presented in this chapter have been submitted in [62], where Xiao He was mostly responsible for the development of the main concept, implemented the main algorithms and wrote the largest parts of the paper; Christian Böhm supervised the project and contributed to the mathematic concept of PIM; Linfei Zhou helped with the algorithm implementation; Jing Feng performed part of the experiments;

Claudia Plant revised the whole paper; The co-authors also contributed to the conceptual development and paper writing.

> *"Xiao He, Linfei Zhou, Jing Feng, Claudia Plant and Christian Böhm. A Probabilistic Integral Metric for Multi-instance Objects. Submitted for publication."*

The generative model of PIM is based on the idea that each multi-instance object is a manifestation of some template defining the major patterns of the data distribution in the instance space. In contrast to existing similarity measures, PIM has a sound probabilistic foundation and combines the following benefits: (1) PIM is a metric which is essential for effective data mining and efficient similarity search. Many data mining techniques like clustering or embedding approaches require metric properties. Metric index structures like the M-tree are ready to speed up similarity search with PIM. (2) Our experiments demonstrate that PIM is a highly effective metric providing superior similarity query and accurate classification and yielding a high index selectivity. (3) PIM is scalable to large data consisting of a high number of multi-instance objects represented by massive amounts of instances.

The remainder of this chapter is organized as follows: In Section 6.1, it starts with an introduction. We give our similarity definition PIM in Section 6.2. Section 6.4.2 analysis the efficiency and index support of PIM. Section 6.4 contains an extensive experimental evaluation. Section 6.5 briefly discusses related work and Section 6.6 concludes the chapter.

6.1 Introduction

Multi-instance objects have become important in database research areas like similarity search and data analysis. In application domains like life sciences, multimedia retrieval, and statistical analysis it is important, to represent each object O_i of similarity search not only by a fixed number of attributes (a single feature vector $\mathbf{x}_i \in \mathbb{R}^d$) but by a *finite set of feature vectors* $\mathscr{X}_i = \{\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,n_i}\} \in \mathscr{P}(\mathbb{R}^d)$ each of which is an element of a multi-dimensional feature space. Thus each of these multi-instance objects is an element of the power set

$\mathscr{P}(\mathbb{R}^d)$ of the d-dimensional feature space. For instance, the performance statistics of an athlete can be naturally captured as multi-instance data: Each athlete, for example an NBA player, is an object which is represented by multiple instances representing the games he/she played. Each instance is a feature vector composed of various performance statistics, including, e.g., the number of points, rebounds, assists, steals, etc.

6.1.1 Motivation

To be able to search for similar objects and to do some analysis on the basis of similarity, we need a similarity measure (actually in most cases a *dis*similarity measure) which assigns to each pair of multi-instance objects a score value indicating the degree of (dis)similarity. Many proposals have been made like the sum of minimum distances (SMD, [92]), the Hausdorff distance (HD, [41]), and others.
Most of these dissimilarity measures are *distance functions*

$$d : \mathscr{P}(\mathbb{R}^d) \times \mathscr{P}(\mathbb{R}^d) \to \mathbb{R}_0^+,$$

meaning that they are, for all objects $\mathscr{X}, \mathscr{Y}, \mathscr{Z} \in \mathscr{P}(\mathbb{R}^d)$:

(M_1) **positive definite**: $d(\mathscr{X}, \mathscr{Y}) = 0 \Leftrightarrow \mathscr{X} = \mathscr{Y}$

(M_2) **symmetric**: $d(\mathscr{X}, \mathscr{Y}) = d(\mathscr{Y}, \mathscr{X})$

A distance function is called a *metric* if it additionally fulfills:

(M_3) **triangle inequality**: $d(\mathscr{X}, \mathscr{Z}) \leq d(\mathscr{X}, \mathscr{Y}) + d(\mathscr{Y}, \mathscr{Z})$

For instance, SMD (sum of minimum distances) is defined:

$$d_{\mathrm{SMD}}(\mathscr{X}, \mathscr{Y}) = \frac{1}{|\mathscr{X}|+|\mathscr{Y}|}\Big(\sum_{\mathbf{x}\in\mathscr{X}} \min_{\mathbf{y}\in\mathscr{Y}} |\mathbf{x}-\mathbf{y}| + \sum_{\mathbf{y}\in\mathscr{Y}} \min_{\mathbf{x}\in\mathscr{X}} |\mathbf{x}-\mathbf{y}|\Big)$$

where $|\mathbf{x}-\mathbf{y}|$ is the Euclidean distance of the vectors. SMD is a distance function but not a metric.

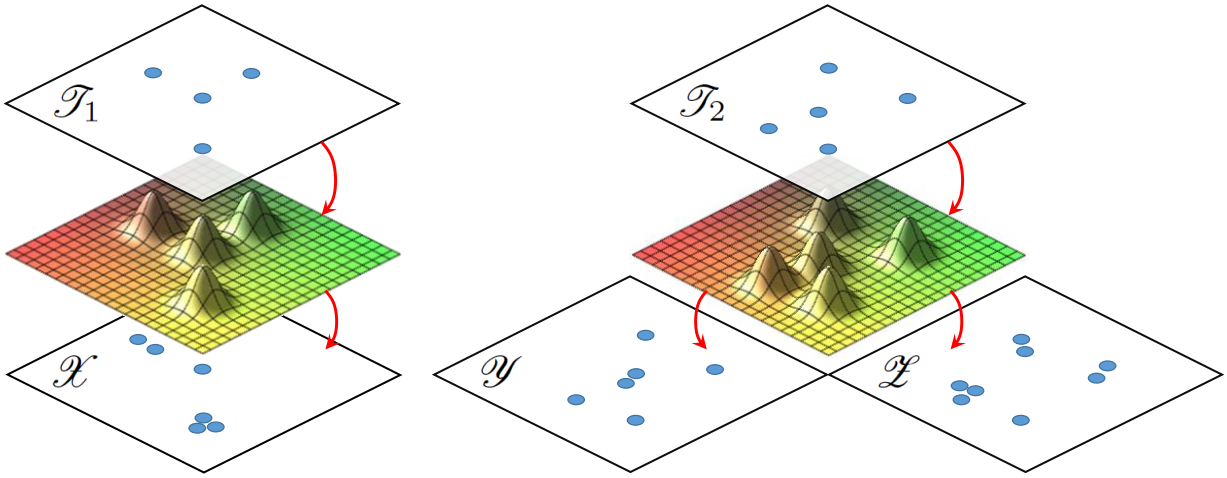


Figure 6.1: Three multi-instance objects $(\mathscr{X}, \mathscr{Y}, \mathscr{Z})$, derived from two templates $(\mathscr{T}_1, \mathscr{T}_2)$ through a generative model.

The properties $(M_1), ..., (M_3)$ are particularly important because they allow us to store the multi-instance objects in an indexing structure for metric objects (like the M-tree [33], GNAT [27] and others). These structures facilitate queries for similar database objects in sublinear time by exploiting (M_3). Moreover, some analysis techniques like clustering and embedding (e.g. DBSCAN [42], FASTMAP [43], Multidimensional Scaling [74]) either fully require the properties of a metric (and are not at all applicable to non-metric distance functions), or become more efficient in runtime and/or more reliable in the result quality by the properties of a metric. If a similarity measure is not a metric, the only way to guarantee the runtime efficiency and the applicability of certain analysis techniques is to propose specialized indexing and analysis techniques just for the given application. So having a *metric* as a similarity measure prevents us from re-inventing the wheel for every new application.

A second example of a well-known multi-instance similarity measure is the Hausdorff-distance (HD), defined as follows:

$$d_{\mathrm{HD}(\mathscr{X},\mathscr{Y})} = \max\left\{ \max_{\mathbf{x}\in\mathscr{X}} \min_{\mathbf{y}\in\mathscr{Y}} |\mathbf{x}-\mathbf{y}|, \max_{\mathbf{y}\in\mathscr{Y}} \min_{\mathbf{x}\in\mathscr{X}} |\mathbf{x}-\mathbf{y}| \right\}$$

This similarity measure happens to be a metric, and therefore, it has become very popular.

However, HD appears to be unsuitable for many applications due to its locality characteristic: On the one hand, every instance $\mathbf{x} \in \mathscr{X}$ is compared to its nearest counterpart in $\mathbf{y} \in \mathscr{Y}$ such that $|\mathbf{x} - \mathbf{y}| = \min$ (and vice versa) which makes the similarity measure good and intuitive. On the other hand, we have a problem if at least one of the instances of $\mathscr{X}$ or $\mathscr{Y}$ has no good partner in the other set, because the *worst* of these outlier instances determines the overall dissimilarity due to the usage of the maximum function. Maybe all of the instances have a very close partner in the other multi-instance object but this is all not considered in the final similarity value if one instance has no good match. So HD is extremely sensitive to such outlier instances which are often understood as *insertions* and *deletions* of instances in this context.

In the obvious solution of this problem, to replace the maximum by the minimum, called the *minimum Hausdorff distance*, unfortunately the metric property (M_3) is lost. Moreover, it can happen just by chance that two instances of two objects match well while the majority of instances are not similar at all. A good and robust similarity measure should not rate the overall similarity among two multi-instance objects based on a single pair of instances which may match or mismatch but *use all available information* to a profound rating.

6.1.2 Goal

We identify the following essential requirements for a multi-instance object similarity measure:

- Metric,
- use all available information,
- robust against outlying instances,
- scalable to large data.

Since none of the existing similarity measures meet all these requirements, our goal is to design a novel one. But how to formalize these intuitive desirable properties of a similarity

measure? While the classical way is just to define a similarity measure by a mathematical formula like that of HD or SMD and then to state the properties and prove them, we believe that a better alternative is first to formalize a generative model which describes the stochastic processes that generate the multi-instance objects, and then to derive the actual similarity model from the generative model. The advantage of this approach is that the generative model is closer to the intuition and that it is easier to assess the quality and properties of the similarity measure and to judge if the corresponding similarity measure is suitable for the similarity search problem at hand. Thus, the generative model gives us a more principled transition from the intuition of the similarity measure to its formal definition.

6.1.3 Idea of our Technique PIM

The generative model underlying the similarity measure of this chapter is visualized in Figure 6.1. We assume that the multi-instance objects in our database are manifestations of some unknown template multi-instance objects, or simply templates. A template is a multi-instance object where we call the instances template instances. Typically, a few templates generate all the multi-instance objects in the database as follows: each instance of a multi-instance object is a manifestation of a template instance with some associated randomness as a consequence of measurement errors, noise etc. One template instance may generate several object instances. Some template instances may also not generate any manifestation in some of the objects. Two similar multi-instance objects have been generated using a common template. Note that this does not mean that we have a specific assumption (like Gaussianity) about the distribution of instances of the whole multi-instance object (which could be too prohibitive for some applications). We only assume that the error of a template instance follows some probability distribution, defined by a variance parameter σ^2.

Revisiting our NBA example, this generative model perfectly makes sense: as the basketball coach Alex Roberts notes, each position of the game (point guard, center, shooting

guard, small forward, power forward) ideally requires a specialized prototype player with specific skills [1]. The prototype players can be regarded as templates, each exhibiting a unique performance profile of template instances. The actual players and their performance statistics are manifestations of their template, e.g. the center Dwight Howard and the shooting guard Kobe Bryant. If two players are very similar according to our metric they are manifestations of a common template, e.g. both shooting guards. For team building, dissimilar players should be combined.

From our generative model, we derive our new similarity measure, called Probabilistic Integral Metric (PIM). Whenever we determine the distance between two multi-instance objects $\mathscr{X}$ and $\mathscr{Y}$, we only know the manifestations, not the templates from which they have been generated. Therefore, our idea is to assume a possible template instance (and thus a generating Gaussian function) at every position $\mathbf{t}$ of the feature space $\mathbb{R}^d$. For a fixed Gaussian centered by $\mathbf{t}$ we can determine the likelihood with which instances of $\mathscr{X}$ and $\mathscr{Y}$ have been generated. But we can also vary $\mathbf{t}$ and sum up all these likelihoods by an integral, which is interestingly always finite. We will prove that the resulting similarity measure fulfills $(M_1), ..., (M_3)$ and thus is indeed a metric.

6.1.4 Contributions

In contrast to established similarity measures, the Probabilistic Integral Metric (PIM) combines all the following benefits:

- **Metric.** In contrast to many other similarity measures (like SMD, RIBL, AL, etc.) our new method PIM is a metric which is essential for effective data mining and efficient similarity search.

- **Probabilistic foundation.** We base our similarity measure on a probabilistic generative model for multi-instance data. This model integrates all available information to evaluate the similarity among multi-instance objects and is robust against outlying

[1] `http://bleacherreport.com/articles/174989-how-to-build-a-championship-basketball-team`

instances. Our model introduced in Section 6.2 requires only few simple and very general assumptions about the probability distribution of the instances and gives a deeper understanding of the properties of our similarity measure.

- **Effectiveness: Superior similarity query and accurate classification.** Our experiments in Section 6.4 comparing our metric to Hausdorff, SMD, Quantile distance and the very expensive Netflow distance demonstrate that PIM is the best choice for similarity query in multi-instance data.

- **Efficiency: Scalable to data with a high number of instances.** Among all comparison methods, our metric is the only similarity measure which scales linearly in the number of instances and thus clearly is the best choice or even the only applicable method in applications involving a high number of instances.

6.2 Probabilistic Integral Metric

Our application defines a d-dimensional feature space $\mathbb{R}^d$ which describes the various properties of our data objects. While a classical (single-instance) object is represented by a single feature vector $\mathbf{x}_i \in \mathbb{R}^d$, a multi-instance object is represented by a *finite set* of feature vectors $\mathscr{X}_i = \{\mathbf{x}_1, \dots \mathbf{x}_{n_i}\}$, where $n_i = |\mathscr{X}_i|$ is the cardinality of the set. Since a multi-instance object is a set of feature vectors, the multi-instance object is an element of the power set $\mathscr{P}(\mathbb{R}^d)$ (while the power set $\mathscr{P}(\mathbb{R}^d)$ contains all (finite *and* infinite) subsets of $\mathbb{R}^d$, usually only the finite subsets are used as multi-instance objects). Following the usual mathematical conventions, we will always denote vectors with $\mathbf{x}, \mathbf{y}, \mathbf{z}$, the elements of vector $\mathbf{x}$ with $(x_1, \dots, x_d)$, and sets with $\mathscr{X}, \mathscr{Y}, \mathscr{Z}$. Our database $\mathscr{D}$ consists of n multi-instance objects $\mathscr{D} = \{\mathscr{X}_1, \dots, \mathscr{X}_n\} \subset \mathscr{P}(\mathbb{R}^d)$. The task of our metric is to identify objects which are similar to a query object $\mathscr{Y} \in \mathscr{P}(\mathbb{R}^d)$ which could be defined by a threshold $\epsilon \in \mathbb{R}_0^+$ for the similarity:

$$\left\{ \mathscr{X} \in \mathscr{D} \;\middle|\; d(\mathscr{X}, \mathscr{Y}) \leq \epsilon \right\} \qquad \text{(Range- Query)}$$

A further possibility is the Nearest Neighbor Query:

$$\left\{ \mathscr{X} \in \mathscr{D} \;\middle|\; d(\mathscr{X}, \mathscr{Y}) = \min_{\mathscr{Z} \in \mathscr{D}} d(\mathscr{Z}, \mathscr{Y}) \right\}$$

6.2.1 The Generative Model

Before actually defining our new similarity metric PIM, we propose a generative statistical model which clarifies our understanding of how similar and dissimilar multi-instance objects come into existence. We believe (and finally demonstrate in Section 6.4) that the actual, observed multi-instance objects are manifestations of template multi-instance objects and are generated from them through stochastic probability distribution functions. We do not assume that these templates are known to the similarity metric and do not require to estimate or predict these templates but we assume that templates exist and that two *similar* objects have a single template *in common*. Like the manifestations of multi-instance objects, a template $\mathscr{T} \in \mathscr{P}(\mathbb{R}^d)$ is a set of d-dimensional feature vectors. A template represents a specific pattern of instances which establishes a probability distribution function in the instance space.

Definition 7. *(Template Distribution Function)*

A template $\mathscr{T} = \{\mathbf{t}_1, \mathbf{t}_2, \dots\} \in \mathscr{P}(\mathbb{R}^d)$ is a set of vectors. Together with a variance parameter $\sigma^2 \in \mathbb{R}^+$ a template defines the following probability distribution function in $\mathbb{R}^d$:

$$g_{\sigma^2}(\mathbf{v}) = \frac{1}{|\mathscr{T}|} \sum_{\mathbf{t} \in \mathscr{T}} \frac{1}{(2\pi\sigma^2)^{d/2}} \cdot \mathrm{e}^{-\frac{|\mathbf{v}-\mathbf{t}|^2}{2 \cdot \sigma^2}}$$

In Figure 6.1 we can see template $\mathscr{T}_1$ consisting of $|\mathscr{T}_1| = 4$ instances, and template $\mathscr{T}_2$ consisting of $|\mathscr{T}_2| = 5$ instances. Each of these templates define a distribution function shown below the templates. From $\mathscr{T}_1$, the manifestation $\mathscr{X}$ is derived, consisting of six instances. While one instance in the template has not generated any manifestation instance in $\mathscr{X}$, each of the others has generated up to three instances. The template $\mathscr{T}_2$ has two associated multi-instance objects $\mathscr{Y}$ and $\mathscr{Z}$, also with a varying number of instances. We

will see that those objects sharing a common template will be spotted as *similar* by our similarity measure PIM.

6.2.2 Our Similarity Metric

Our similarity measure is motivated by two aspects. Firstly, we aim at maintaining the generative model in order to profit from its properties. Since we assume that the set of templates is unknown to the similarity measure and we do not want to estimate the set of templates (our similarity measure needs only the information of the two objects $\mathscr{X}$ and $\mathscr{Y}$ the similarity of which has to be determined, not that of any other objects), our strategy is to assume a template instance $\mathbf{t}$ at every position of the data space $\mathbb{R}^d$, and to determine the likelihoods of the actual instances under this assumption. These likelihoods are finally summed up using a volume (multidimensional) integral iterating over the whole data space. Secondly, we aim at defining a similarity measure which is a metric and specifically fulfills (M_3), the triangle inequality. To achieve this, we use the Hamming distance as a basis. The Hamming distance, originally designed for vectors of categorical values, corresponds to the number of components in which the two vectors are not equal:

$$d_{\text{Hamming}}(\mathbf{v}, \mathbf{w}) = \sum_{1 \leq i \leq d} \begin{cases} 1 & \text{if } v_i \neq w_i \\ 0 & \text{otherwise.} \end{cases}$$

Since the Hamming distance is a metric, we adopt this idea in the following way: we decompose the feature space into an infinite number of overlapping (hyper-) spheres having varying radius r and center $\mathbf{t}$. We treat each of these spheres from the space decomposition as a categorical variable. If it is covered by an instance of multi-instance object $\mathscr{X}$, then this categorical variable is set to one, otherwise to zero by a function $c_{\mathscr{X}}(\mathbf{t}, r) \in \{0, 1\}$. In our distance measure, the variable is weighted by the probability density of the corresponding radius r, mainly for two reasons: Firstly, we obtain exactly the behavior of our generating model if we apply the Gaussian function with variance σ^2. Secondly, for our similarity measure we can sum up all these weighted differences between the coverages of

the spheres by object $\mathscr{X}$ and $\mathscr{Y}$ by an integral operator, and this integral becomes finite and solvable by this weighting factor.

First of all, we need to derive the distribution function of the *radius* r from the Template Distribution Function, cf. Def. 7. If the likelihood of a vector $\mathbf{v}$ is

$$p(\mathbf{v}) = \frac{1}{(2\pi\sigma^2)^{d/2}} \cdot \mathrm{e}^{-\frac{|\mathbf{v}-\mathbf{t}|^2}{2\cdot\sigma^2}}$$

then the likelihood that the distance between $\mathbf{v}$ and $\mathbf{t}$ equals r can be determined by multiplying this formula with the surface of the sphere $\pi^{d/2} \cdot d/\Gamma(\frac{d}{2}-1) \cdot r^{d-1}$ with $\Gamma(1) = 1, \Gamma(\frac{1}{2}) = \sqrt{\pi}, \Gamma(x+1) = x \cdot \Gamma(x)$. Note that the sphere is centered by $\mathbf{t}$ and all vectors $\mathbf{v}$ with the same distance from $\mathbf{t}$ have the same likelihood. The probability density function of the radius r is:

$$p(r) = \frac{d \cdot r^{d-1}}{(2\sigma^2)^{d/2}\Gamma(\frac{d}{2}+1)} \cdot \mathrm{e}^{-\frac{r^2}{2\cdot\sigma^2}}$$

Using $p(r)$, our dissimilarity measure is defined as follows:

Definition 8. *(Probabilistic Integral Metric)*

$$d_{PIM}(\mathscr{X}, \mathscr{Y}) = \int_{\mathbb{R}^d} \int_0^\infty p(r) \cdot h_{\mathscr{X},\mathscr{Y}}(\boldsymbol{t}, r)\ dr\ d\boldsymbol{t}.$$

where $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r)$ tells us if the circle centered by $\mathbf{t}$ with radius r is *differently* covered by instances from $\mathscr{X}$ and $\mathscr{Y}$:

$$h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) = \begin{cases} 1 & \text{if } c_{\mathscr{X}}(\mathbf{t}, r) \neq c_{\mathscr{Y}}(\mathbf{t}, r) \\ 0 & \text{otherwise.} \end{cases}$$

$$c_{\mathscr{X}}(\mathbf{t}, r) = \begin{cases} 1 & \text{if } \min_{\mathbf{x}\in\mathscr{X}} |\mathbf{t}-\mathbf{x}| \leq r \\ 0 & \text{otherwise.} \end{cases}$$

The volume integral of some function $f(\mathbf{v})$ stands for:

$$\int_{\mathbb{R}^d} f(\mathbf{v})\ \mathrm{d}\mathbf{v} = \int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} f(\mathbf{v})\ \mathrm{d}v_1 \dots \mathrm{d}v_d.$$

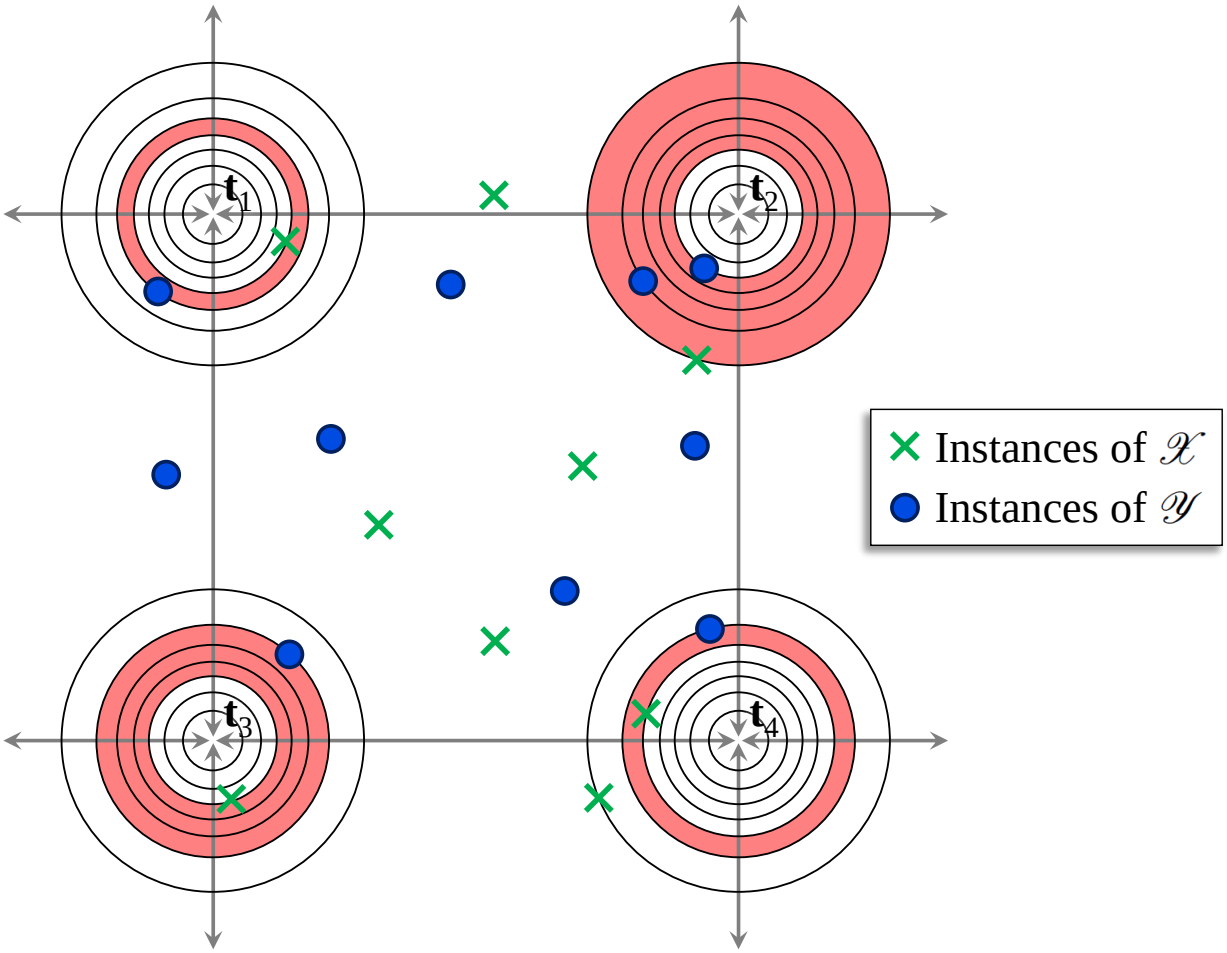


Figure 6.2: Intuition of Definition 8: The template instance $\mathbf{t}$ is varied over the whole data space $\mathbb{R}^{d=2}$. Shown are four examples $(\mathbf{t}_1, \ldots, \mathbf{t}_4)$ with their Gaussian probability density functions and with those circles colored in red having $c_{\mathcal{X}}(\mathbf{t}_i, r) \neq c_{\mathcal{Y}}(\mathbf{t}_i, r)$ ("differently covered").

A cell is differently covered if it contains at least one instance of object $\mathcal{X}$ and no instances of the other object $\mathcal{Y}$ (or vice versa). In Definition 8, it is sufficient to always apply only that instance $\mathbf{x}$ which has minimal distance to the center of the cell $\mathbf{t}$ since we integrate over all possible centers $\mathbf{t}$ and all possible radii r.

The intuition of PIM is visualized in the 2D example of Figure 6.2 where we have the instances of $\mathcal{X}$ marked by crosses and the instances of $\mathcal{Y}$ marked by circles, respectively. The template Gaussian is actually moved over the whole data space from $-\infty$ to $+\infty$ in all dimensions, but shown are only 4 sample instances together with their probability

density functions. Each of these Gaussians consists of an infinite number of concentric circles (hyper-spheres in the general-dimensional case of $d > 2$). We have marked those circles in red color that are *differently covered* according to our definition of function $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r)$. For instance in the upper right Gaussian centered by $\mathbf{t}_2$, the area of marked circles starts with the second innermost drawn circle (covered by an instance of $\mathscr{Y}$) and ends at the outermost drawn circle (covered by an instance of $\mathscr{X}$). Note that for our notion of *differently covered* we do not count the exact number of instances of $\mathscr{X}$ and $\mathscr{Y}$, respectively, but only determine *if* the circle is covered at least by one instance.

Next, we give the proof that PIM fulfills the properties (M_1), (M_2), and (M_3) of a metric, before refining PIM in the next section for an efficient computation of $d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y})$.

Lemma 1. *PIM is a metric*

Proof. (M_1) **Positive Definiteness:**

The integrated function is everywhere ≥ 0. If the instances of $\mathscr{X}$ and $\mathscr{Y}$ are exactly equal, then none of the spheres are differently covered. Therefore $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) = 0$ for all $\mathbf{t}$ and r, and thus $d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y}) = 0$. If some instances are different then there exist some spheres which are differently covered. In this case these spheres are multiplied with some positive factor, and have thus a positive influence on the integral. Therefore, in this case, $d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y}) > 0$.

(M_2) **Symmetry:**

From the definition of $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r)$ we can see:

$$h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) = h_{\mathscr{Y},\mathscr{X}}(\mathbf{t}, r).$$

As $p(r)$ is independent of $\mathscr{X}$ and $\mathscr{Y}$, we obtain also

$$\int_{\mathbb{R}^d}\int_0^\infty p(r)\, h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r)\,\mathrm{d}r\,\mathrm{d}\mathbf{t} = \int_{\mathbb{R}^d}\int_0^\infty p(r)\, h_{\mathscr{Y},\mathscr{X}}(\mathbf{t}, r)\,\mathrm{d}r\,\mathrm{d}\mathbf{t}$$
$$d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y}) = d_{\mathrm{PIM}}(\mathscr{Y}, \mathscr{X})$$

(M_3) **Triangle Inequality:**

We have to show that

$$d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Z}) \leq d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y}) + d_{\mathrm{PIM}}(\mathscr{Y}, \mathscr{Z})$$

always holds. First, we show an analogous property for $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r)$ by a case analysis on $c_{\mathscr{X}}(\mathbf{t}, r) \in \{0, 1\}$. We can distinguish only the following four cases:

1. $c_{\mathscr{X}}(\mathbf{t}, r) = c_{\mathscr{Y}}(\mathbf{t}, r) = c_{\mathscr{Z}}(\mathbf{t}, r)$
   $h_{\mathscr{X},\mathscr{Z}}(\mathbf{t}, r) = 0 \quad \leq \quad 0 + 0 = h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) + h_{\mathscr{Y},\mathscr{Z}}(\mathbf{t}, r)$

2. $c_{\mathscr{X}}(\mathbf{t}, r) = c_{\mathscr{Y}}(\mathbf{t}, r) \neq c_{\mathscr{Z}}(\mathbf{t}, r)$
   $h_{\mathscr{X},\mathscr{Z}}(\mathbf{t}, r) = 1 \quad \leq \quad 0 + 1 = h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) + h_{\mathscr{Y},\mathscr{Z}}(\mathbf{t}, r)$

3. $c_{\mathscr{X}}(\mathbf{t}, r) = c_{\mathscr{Z}}(\mathbf{t}, r) \neq c_{\mathscr{Y}}(\mathbf{t}, r)$
   $h_{\mathscr{X},\mathscr{Z}}(\mathbf{t}, r) = 0 \quad \leq \quad 1 + 1 = h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) + h_{\mathscr{Y},\mathscr{Z}}(\mathbf{t}, r)$

4. $c_{\mathscr{X}}(\mathbf{t}, r) \neq c_{\mathscr{Y}}(\mathbf{t}, r) = c_{\mathscr{Z}}(\mathbf{t}, r)$
   $h_{\mathscr{X},\mathscr{Z}}(\mathbf{t}, r) = 1 \quad \leq \quad 1 + 0 = h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) + h_{\mathscr{Y},\mathscr{Z}}(\mathbf{t}, r)$

Since $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r)$ fulfills the triangle inequality and $p(r) > 0$ for all r and independent of $\mathscr{X}, \mathscr{Y}, \mathscr{Z}$, we know also that

$$p(r) \cdot h_{\mathscr{X},\mathscr{Z}}(\mathbf{t}, r) \leq p(r) \cdot h_{\mathscr{X},\mathscr{Y}}(\mathbf{t}, r) + p(r) \cdot h_{\mathscr{Y},\mathscr{Z}}(\mathbf{t}, r).$$

For the Riemann integral the following monotonicity is valid:

$$\text{If } \forall \mathbf{v} \in \mathbb{R}^d : f_1(\mathbf{v}) \leq f_2(\mathbf{v}) \text{ then } \int_{\mathbb{R}^d} f_1(\mathbf{v}) \, \mathrm{d}\mathbf{v} \leq \int_{\mathbb{R}^d} f_2(\mathbf{v}) \, \mathrm{d}\mathbf{v}$$

and analogously for $\int_0^\infty \ldots \mathrm{d}r$. Thus, we obtain

$$\begin{aligned}\int_{\mathbb{R}^d}\int_0^\infty p(r)\, h_{\mathscr{X},\mathscr{Z}}(\mathbf{t},r)\,\mathrm{d}r\,\mathrm{d}\mathbf{t} \leq \int_{\mathbb{R}^d}\int_0^\infty p(r)\, h_{\mathscr{X},\mathscr{Y}}(\mathbf{t},r)\,\mathrm{d}r\,\mathrm{d}\mathbf{t} + \\ + \int_{\mathbb{R}^d}\int_0^\infty p(r)\, h_{\mathscr{Y},\mathscr{Z}}(\mathbf{t},r)\,\mathrm{d}r\,\mathrm{d}\mathbf{t} \\ d_{\mathrm{PIM}}(\mathscr{X},\mathscr{Z}) \leq d_{\mathrm{PIM}}(\mathscr{X},\mathscr{Y}) + d_{\mathrm{PIM}}(\mathscr{Y},\mathscr{Z})\end{aligned}$$

□

6.2.3 Efficient Evaluation of PIM

How to simplify the formula given in Def. 8 and how to restrict it to local neighbors of some instances in $\mathscr{X}$ and $\mathscr{Y}$ only? The inner integral of $d_{\mathrm{PIM}}(\mathscr{X},\mathscr{Y})$ varies from 0 to ∞, but as we can see from Figure 5.2, $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t},r) = 0$ for all $r \leq r_{\min}$ and for all $r \geq r_{\max}$ where $r_{\min}$ is the smallest distance of $\mathbf{t}$ to any instance of $\mathscr{X} \cup \mathscr{Y}$. If $r_{\min}$ is determined by an instance of $\mathscr{X}$, then $r_{\max}$ is the smallest distance of $\mathbf{t}$ to an instance of $\mathscr{Y}$ and vice versa:

$$\begin{aligned} r_{\min} &= \min\left\{\min_{\mathbf{x}\in\mathscr{X}} |\mathbf{x}-\mathbf{t}|, \min_{\mathbf{y}\in\mathscr{Y}} |\mathbf{y}-\mathbf{t}|\right\} \\ r_{\max} &= \max\left\{\min_{\mathbf{x}\in\mathscr{X}} |\mathbf{x}-\mathbf{t}|, \min_{\mathbf{y}\in\mathscr{Y}} |\mathbf{y}-\mathbf{t}|\right\}\end{aligned}$$

Therefore, we can adapt the integration boundaries instead of applying $h_{\mathscr{X},\mathscr{Y}}(\mathbf{t},r)$:

$$d_{\mathrm{PIM}}(\mathscr{X},\mathscr{Y}) = \int_{\mathbb{R}^d}\int_{r_{\min}}^{r_{\max}} p(r)\ \mathrm{d}r\ \mathrm{d}\mathbf{t}.$$

The cumulative distribution function $P(r) = \int_0^r p(r')\ \mathrm{d}r'$ can be determined as follows:

$$\begin{aligned} P(r) &= \int_0^r \frac{d\cdot(r')^{d-1}}{(2\sigma^2)^{d/2}\Gamma(\frac{d}{2}+1)} \cdot \mathrm{e}^{-\frac{(r')^2}{2\cdot\sigma^2}} dr' \\ &= C_1 \cdot \int_0^r (r')^{d-1} e^{-\frac{(r')^2}{2\cdot\sigma^2}} dr'\end{aligned}$$

Let $x = \frac{(r')^2}{2 \cdot \sigma^2}$ and $s = d/2$, then

$$P(r) = C_2 \cdot \int_0^{\frac{r^2}{2 \cdot \sigma^2}} x^{s-1} e^{-x} dx = C_2 \cdot \gamma(s, \frac{r^2}{2 \cdot \sigma^2})$$

where C_1 and C_2 are constants that can be safely left out and $\gamma(s, z)$ is the lower incomplete gamma function which can be solved recursively as:

$$\begin{cases} \gamma(s, z) = (s-1) \cdot \gamma(s-1, z) - z^{s-1} \cdot e^{-z} \\ \gamma(1, z) = 1 - e^{-z} \\ \gamma(\frac{1}{2}, z) = \sqrt{\pi} \cdot erf(\sqrt{z}) \end{cases} \tag{6.1}$$

For our similarity measure, we obtain:

$$d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y}) = \int_{\mathbb{R}^d} P(r_{\max}) - P(r_{\min}) \, \mathrm{d}\mathbf{t}. \tag{6.2}$$

To determine for a given vector $\mathbf{t}$ those instances $\mathbf{x} \in \mathscr{X}, \mathbf{y} \in \mathscr{Y}$ which are responsible for $r_{\max}$ and $r_{\min}$, we need to find the nearest neighbor of $\mathbf{t}$ in $\mathscr{X}$ and $\mathscr{Y}$, respectively. The locus of all points of the feature space which have a common nearest neighbor is determined by the Voronoi diagram. Figure 6.3 shows us the overlay of the Voronoi tessellations of $\mathscr{X}$ (in lite green lines) and $\mathscr{Y}$ (in dark blue lines). The overlay defines a number of at most $|\mathscr{X}| \cdot |\mathscr{Y}|$ convex combination cells in which the same pair of instances is responsible for the values of $P(r_{\max})$ and $P(r_{\min})$. As the two instances change their roles with respect to $r_{\max}$ and $r_{\min}$ at the orthogonal line (plane) in the middle of the two instances, we have to cut some of the convex combination cells into two pieces, as depicted in Figure 6.4. We can further see, that for each of the pieces, we determine the integral $P(r_{\max})$ and $P(r_{\min})$. These are summed up for all pieces of all convex combination cells. This observation proves also that the result of $d_{\mathrm{PIM}}(\mathscr{X}, \mathscr{Y})$ is always finite because it is the sum of a finite number of Gaussian integrals (each of which is finite).

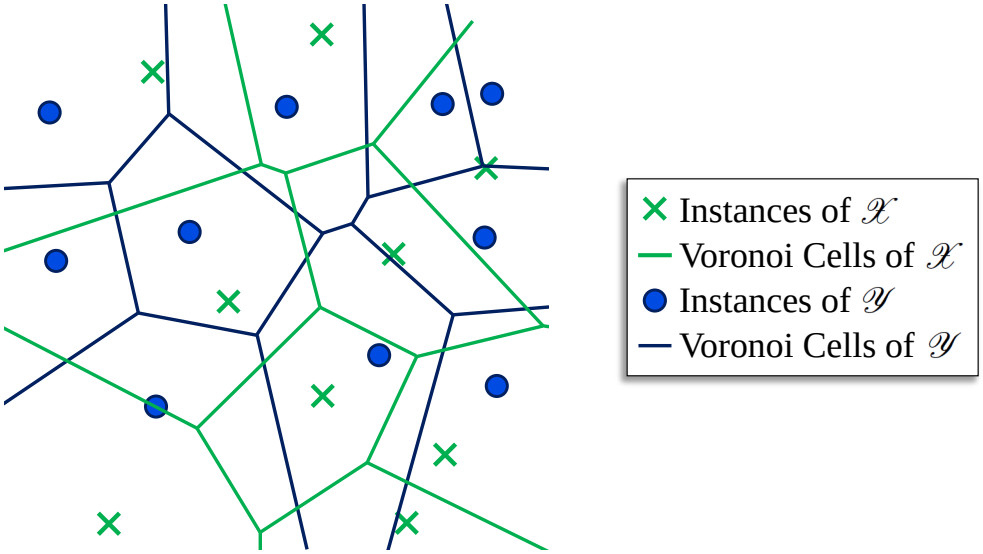


Figure 6.3: Overlay of the Voronoi diagrams of multi-instance object $\mathscr{X}$ and $\mathscr{Y}$.

6.2.4 Monte Carlo Integration

We use Equation (6.2) in a Monte Carlo integration approach. We generate random samples $\mathbf{s} \in \mathbb{R}^d$ from an independent uniform distribution (in a suitable interval). For each sample, we determine the nearest neighbors in $\mathscr{X}$ and $\mathscr{Y}$, respectively, then $r_{\max}$ and $r_{\min}$ and estimate thus the average value of the function $P(r_{\max}) - P(r_{\min})$ to be integrated. This average value times the volume of the interval from which the samples have been taken gives us an accurate approximation of $\text{dist}(\mathscr{X}, \mathscr{Y})$.

The sample range of integration is determined from the statistics of object $\mathscr{X}$ and $\mathscr{Y}$. We calculate for each dimension i the minimum (maximum) instance coordinate:

$$\check{x}_i = \min_{\mathbf{x} \in \mathscr{X} \cup \mathscr{Y}} x_i - \sigma \qquad \hat{x}_i = \max_{\mathbf{x} \in \mathscr{X} \cup \mathscr{Y}} x_i + \sigma$$

In each dimension i, the random samples are taken from the interval $[\check{x}_i, \hat{x}_i]$. In the following, we denote the number of samples used for Monte Carlo integration by S.

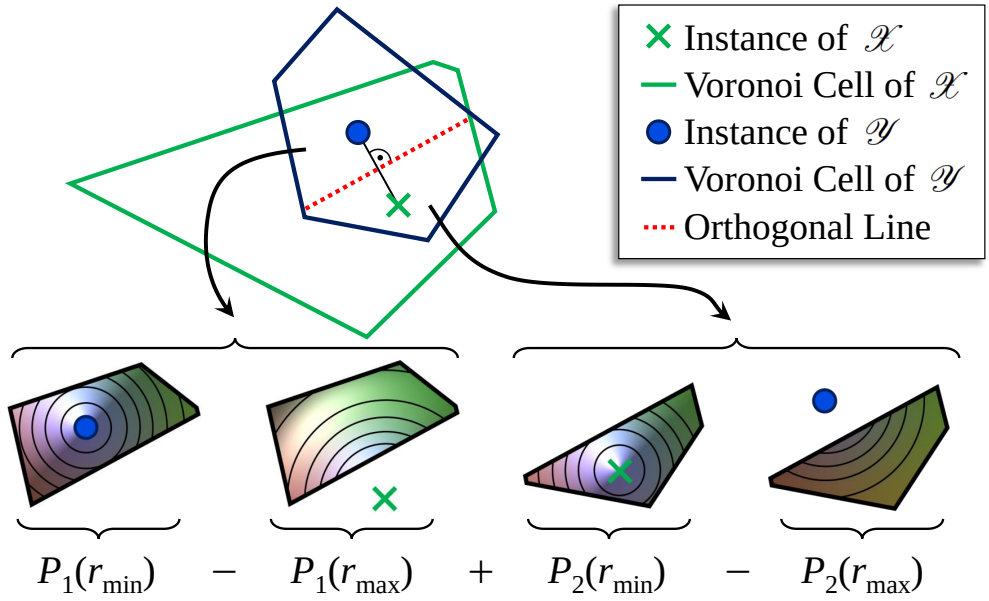


Figure 6.4: A convex combination cell is cut into two pieces by the orthogonal line in the middle between the instances.

6.3 Complexity and Index Support

If we consider two multi-instance objects $\mathscr{X}$ and $\mathscr{Y}$, computing $d_{\mathrm{PIM}}(\mathscr{X},\mathscr{Y})$ requires $S\cdot(|\mathscr{X}|+|\mathscr{Y}|)$ Euclidean distance calculations in $\mathbb{R}^d$. For each sample point, we just need to identify the closest instance in both multi-instance objects, see Section 6.2.4. Obviously, PIM is linear in terms of number of samples. In the Experiment section, we will show that hundreds of samples are enough for PIM to achieve satisfied quality. Furthermore, PIM scales linearly regarding the number of instances. This is a great improvement compared to the existing similarity measures. The best scalable ones own the runtime complexity of $|\mathscr{X}|\cdot|\mathscr{Y}|$, e.g. Hausdorff distance (HD [41]), sum of minimum distance (SMD, [92]) and quantile distance (QD, [118]). There are other more complex similarity measures, like Netflow distance [96]. Netflow is obtained by solving a minimum cost maximum flow problem, which usually needs $|\mathscr{X}|\cdot|\mathscr{Y}|\cdot(|\mathscr{X}|+|\mathscr{Y}|)$ calculations.

Additionally, in applications involving massive amounts of instances, we could speed up PIM in instance space by a multivariate index structure, e.g. the X-tree [19] reducing the runtime complexity of PIM to $O(S\cdot(\log|\mathscr{X}|+\log|\mathscr{Y}|))$. In our experiments, we did

not perform such instance-level indexing, since the total number of instances within the multi-instance objects is not so large in our real data.

Rather, we exploit the metric properties of PIM by performing indexing on the level of the multi-instance objects. More specifically, we organize the multi-instance object in an M-tree [33] which is a hierarchical index structure for metric data. M-tree supports K-nearest neighbor and ϵ-range queries by exploiting the pruning power of the routing objects using the triangle inequality. We use the basic M-tree since this relatively simple index structure allows us to directly assess the effectiveness of our metric.

6.4 Experiments

In this section, we provide extensive experiments on both synthetic and real data sets to show the effectiveness of PIM and the efficiency and scalability of indexing with PIM.

6.4.1 Effectiveness of PIM

Data Sets and Evaluation Methods

Firstly, we generate synthetic data sets to evaluate the effectiveness of PIM. Each data set contains 2 clusters, each with 100 objects and 3 dimensions. Objects in the same cluster share the same center: $[0, 0, 0]$ and $[1, 1, 1]$ for each cluster. The number of instances of an object follows a uniform distribution in range $[1, 2 * 500]$. Locations of instances in an object follow a mixed distribution. They are generated from either *Uniform* or *Normal* distribution. For *Uniform* distribution, instances are located in $[C_i - R, C_i + R]$, where C_i is the center of dimension i. The standard deviation of *normal* distribution is set to $0.2 * R$. Finally, the data set is normalized to $[0, 1]^3$. We generate synthetic data sets by varying R so that objects from different classes may contain different amount of overlapping instances. For each R we generate 10 synthetic data sets and report the average nearest neighbor query accuracy.

Besides, we use two real data sets to evaluate PIM regarding effectiveness: NBA data

and Face image data.

The NBA data is extracted from NBA players' game-by-game statistics from 1986 to 2013 season, containing $621,052$ instances of 2154 players. Each player is regarded as a multi-instance object, where 5 statistics (points, rebounds, assists, steals, and blocks) of a player per game is treated as an instance after normalization. We compare the similar NBA players and NBA team building for evaluation.

The CMU Face Images data set consists of 640 gray scale images of people taken with varying pose, expression and wearing glasses or not. Each image is an object that is smoothed by a Gaussian filter and sub-sampled to 11×11. The 9×9 grid of non-border pixels are chosen as the instances for an image. Each instance contains five features, which are its color and color differences with four neighbors after normalization. CMU Face image data contains class label. Thus we use nearest neighbor accuracy and Multidimensional Scaling to measure the performance of PIM.

In all effectiveness experiments, we compare PIM to the only two metrics between points sets that we know: Hausdorff [41] and Netflow [96]. Besides, we compare PIM to the other two non-metric ones: Sum of Minimum Distances (SMD) [92] and Quantile-distance (QD) [118]. QD requires a parameter $\phi \in (0, 1)$, for the quantile resolution. We experiment QD with $\phi = [0.1, ..., 0.9]$ and report the best one.

Synthetic Data sets

The effects of parameters. The calculation of PIM needs two parameters: number of samples S and σ. Therefore, before comparing PIM to other competitors we firstly evaluate the effects of these two parameters in terms of effectiveness. Specifically, we use nearest neighbor query accuracy to evaluate S and σ on synthetic data sets introduced in Section 6.4.1. S only affects the accuracy of Monte Carlo integration. Thus more samples give more accurate approximation. Therefore, we first fix the number of samples $S = 1000$ and evaluate σ. Then we evaluate S with the best σ.

The left part of Figure 6.5 depicts the nearest neighbor query accuracy of PIM with

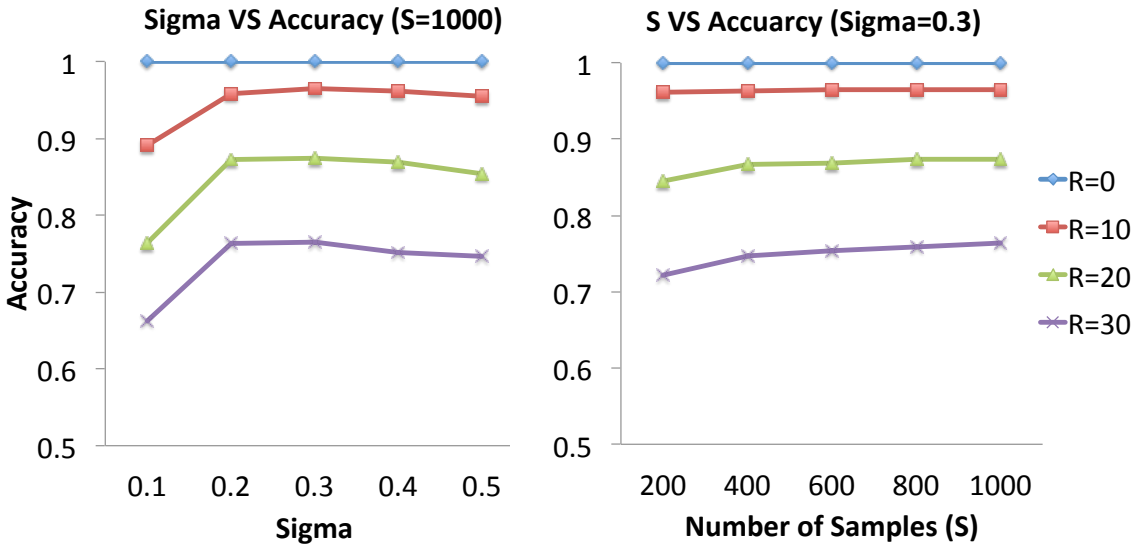


Figure 6.5: Effects of σ and S of PIM on synthetic data sets introduced in Section 6.4.1 with increasing R.

$\sigma = [0.1, ..., 0.5]$ on synthetic data sets. From the figure we can see that PIM performs best with $\sigma = 0.3$ and provides stable results for σ from 0.2 to 0.4 for all the data with different R. Thus, in the following, we will fix $\sigma = 0.3$ for all the data sets. Then, we vary $S = [200, ..., 1k]$. The performance of PIM is shown in right part of Figure 6.5. Generally, the performance of PIM is increasing with S. Larger S gives a better accuracy of PIM, however less efficiency for calculation. In the following, we will evaluate PIM with both $S = 1k$ and $S = 200$ samples.

Comprison with competiors. The synthetic data sets become more difficult for classification with increasing R, since there are more overlapping instances for objects from different classes. We evaluate the performance of PIM compared to competitors on these data sets with increasing R, see Figure 6.6. Generally, all methods degrade their performances with increasing R. Among them PIM and with $S = 1k$ expensive Netflow perform best especially when R is large. With $S = 200$ PIM provides slightly worse results, which are still better than all the others. SMD performs similar with PIM when there are less overlaps. However, it degrades its performance more with increasing R compared to PIM.

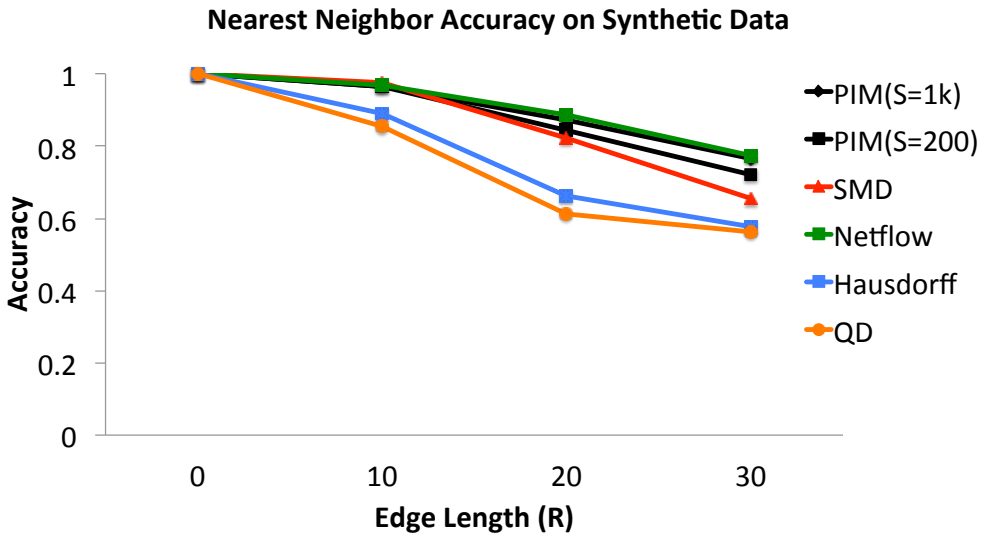


Figure 6.6: Comparison of nearest neighbor accuracy on synthetic data introduced in Section 6.4.1 with increasing R.

Hausdorff and HD perform worst. PIM performs best due to its probability foundation.

NBA Data

K-nearest neighbors of NBA players. We use K-NN queries on NBA data to compare the usefulness of the results and the rankings. For example consider the top 5 similar NBA players of one of most famous NBA player Michael Jordan listed in Table 6.1. PIM with 1k samples provides sensible similar players for Michael Jordan, who is the most successful shooting guard in the NBA history. His first, third and fifth neighbors Clyder Drexler, Dwyane Wade and Kobe Bryant are top shooting guards in NBA history as M. Jordan. The second and fourth neighbor Scotte Pippen and Lebron James are all excellent utility players as M. Jordan. They all have similar performance profiles with M. Jordan: They are top scorers and good at assists and rebounds as well. The performance of PIM using 200 samples is only slightly different: 4 out of 5 nearest neighbors are the same with a slightly different ranking. Hausdorff gives high ranks for Gilbert Arenas and Gary Payton. G. Arenas has outstanding games from 2004 to 2006. However, since 2007 he only played 134 games and his performance degrades a lot. Therefore he is definitely less similar to M. Jordan than D. Wade and K. Bryant. Besides, G. Payton is generally regarded as a

Table 6.1: Top 5 similar players of M. Jordan in NBA data.

PIM($S = 1k$)	PIM($S = 200$)	Hausdorff	Netflow	SMD	QD($\phi = 0.5$)
M. Jordan	M. Jordan	M. Jordan	M. Jordan	M. Jordan	K. Durant
C. Drexler	C. Drexler	C. Drexler	L. James	K. Bryant	L. James
S. Pippen	D. Wade	P. Pierce	A. Iverson	P. Pierce	C. Anthony
D. Wade	P. Pierce	G. Arenas	D. Wade	R. Allen	M. Jordan
L. James	S. Pippen	L. James	K. Bryant	C. Drexler	K. Bryant
K. Bryant	K. Bryant	G. Payton	C. Drexler	V. Carter	D. Wade

better assister than a scorer. This result demonstrates that Hausdorff is heavily influenced by outlying instances. QD provides a strange result since it is not a metric. Three players are ranked before M. Jordan and thus considered to be more similar to M. Jordan than M. Jordan to himself. SMD reports more sensible similarities compared with Hausdorff and QD, because they consider the whole distribution of the instances. Besides C. Drexler and K. Bryant, SMD provides Ray Allen, and Paul Pierce, who are top scorers as well. Netflow also reports reasonable results, e.g. Allen Iverson. Results from PIM, SMD and Netflow share some overlapping players. Comparing which one is better must involve subjective judgment. However, it is clear that PIM, SMD and Netflow can provide more sensible results than Hausdorff and QD on NBA data set, since they use all the available instances information.

NBA team building. As we mentioned in the introduction, another application of our proposed multi-instance metric PIM is sports team building. Taking NBA team building as an example, dissimilar players should be combined for making a better team. We analyze the performance and the diversity of five NBA teams in regular season 2012-2013, cf. Figure 6.7. We selected with Miami Heat (MIA, Eastern Conference Champion) and Oklahoma Thunder (OKC, Western Conference Champion) the two top performing teams. The Atlanta Hawks (ATL) and the Milwaukee Bucks (MIL) are average performers and the Charlotte Bobcats (CHA) finished last in the Eastern Conference. The first column of Figure 6.7 shows the success of the teams in terms of the normalized number of games won. MIA is the best performing team with 66 games won, followed by OKC (60), ATL

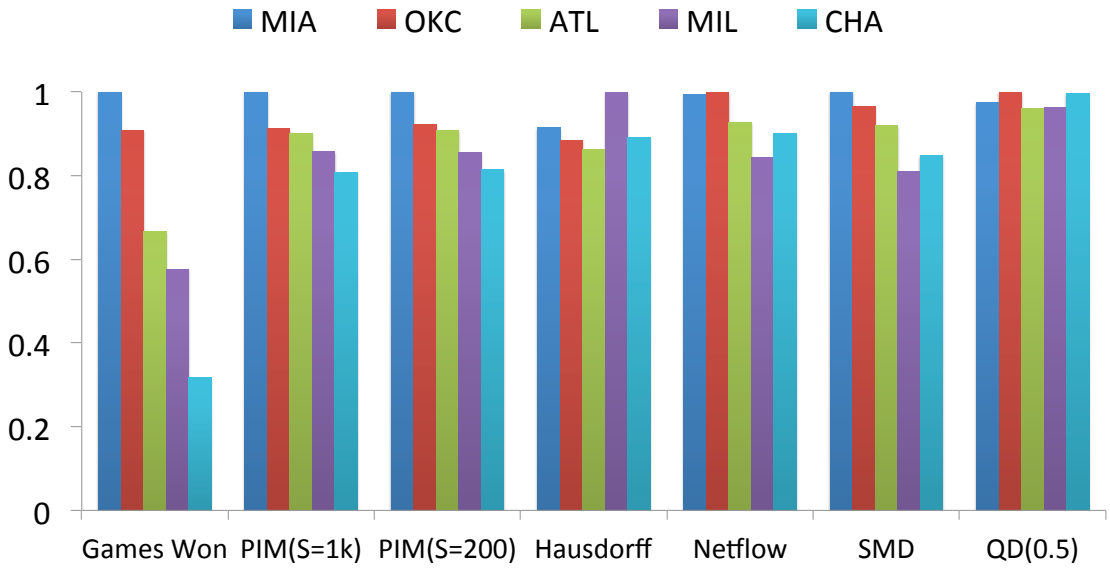


Figure 6.7: Comparing the diversity of NBA teams. First column: Performance of NBA teams provided by the number of games won (normalized). Remaining columns: Diversity in team composition as measured by the average distances among the players (normalized). PIM is the only similarity measure reflecting the ranking in team performance in team diversity as it can be expected from domain knowledge.

(44), MIL (38) and finally CHA with 21 games won. The remaining columns of the figure show the ranking of the teams in terms of diversity as computed by the average distances among players with PIM and the comparison partners (normalized). PIM is the only method reflecting the domain knowledge that better teams are also more diverse teams. With sample size 200 and 1000 the diversity team rankings determined with PIM match the performance ranking in the first column. Hausdorff ranks MIL as the most diverse team. SMD, Netflow and QD rate the best team MIA also as the most diverse team but the worst team CHA as the average diverse team.

CMU Face Data

Nearest Neighbor Accuracy. CMU Face image data contains class label. Thus we firstly use nearest neighbor accuracy to compare PIM with competitors. The results are depicted in Figure 6.8. Netflow and SMD perform best with the accuracy of 0.981. With 1000 and 200 samples PIM performs slightly worse with accuracy of 0.98 and 0.978. Hausdorff gives

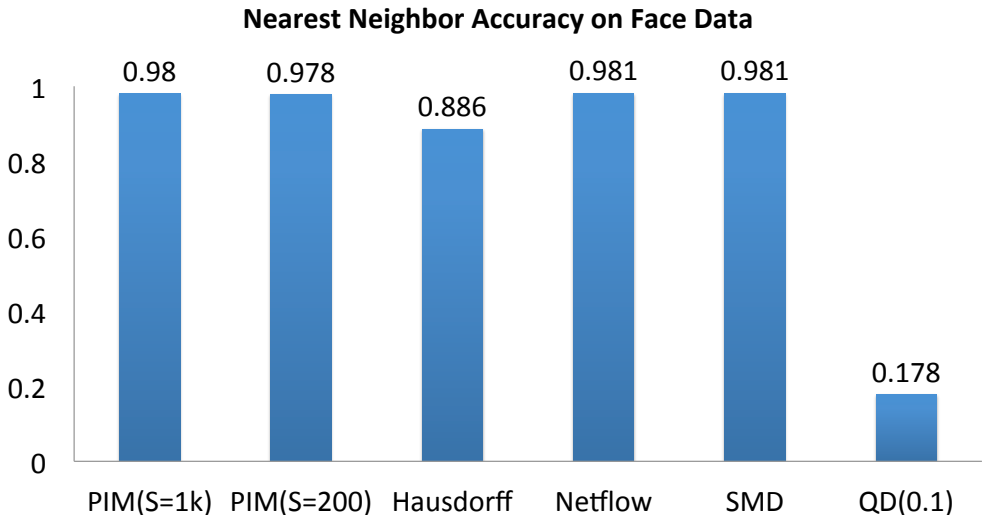
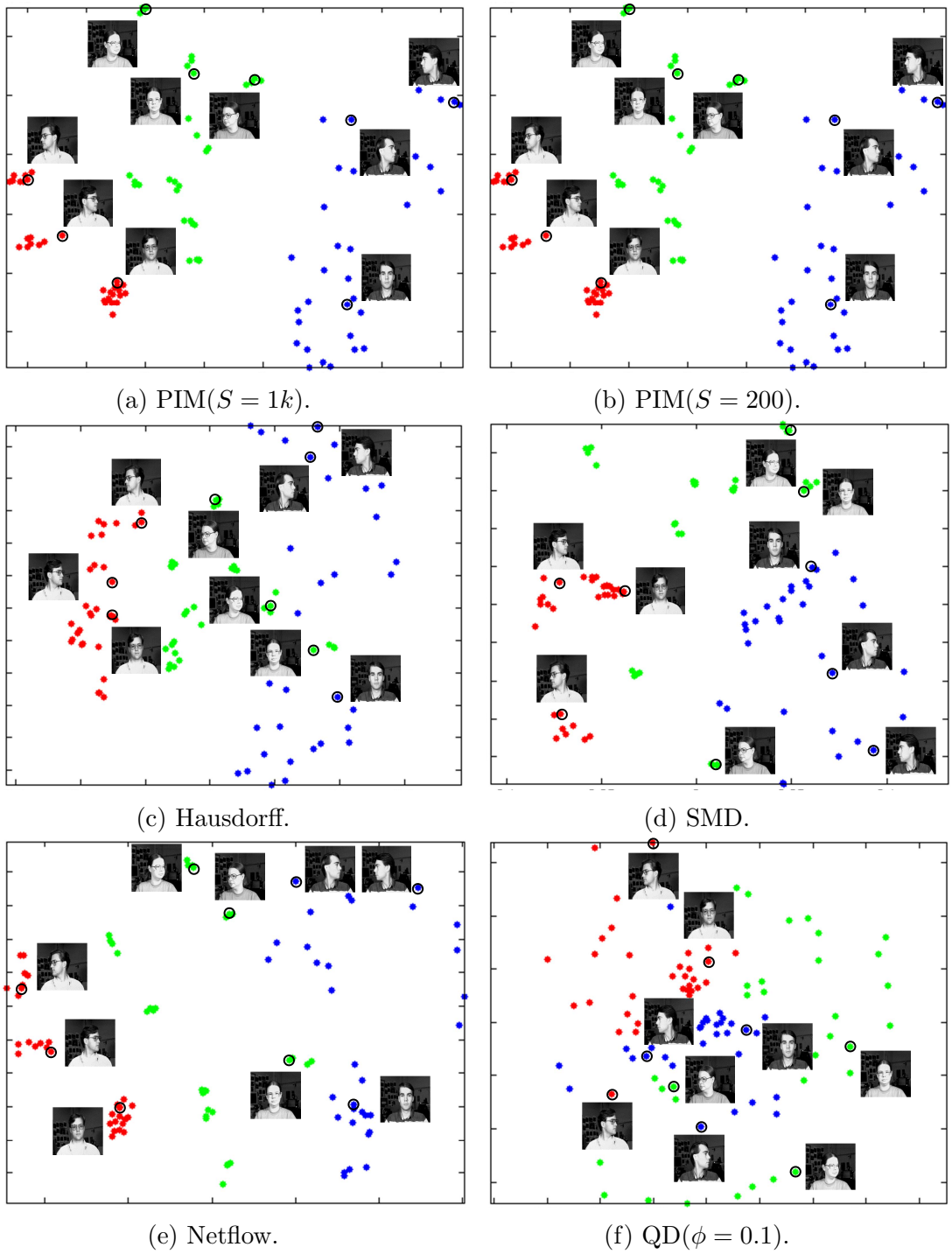


Figure 6.8: Nearest Neighbor Query Accuracy on Face data.

a worse accuracy of around 0.88 and QD provides the worst result. This is because PIM, Netflow and SMD take all the instances information into account, while Hausdorff and QD only consider the extreme instances or part of the instances. This experiment demonstrates the importance of using all the available information to measure the similarity between Multi-instance Objects.

Multidimensional Scaling on Face data. In this part, we use multi-dimensional scaling to evaluate PIM and the competitors. A good metric or distance function should be able to put the faces of the same person closer together than those of other persons. To facilitate visualization and comparison, we project the data to 2D space. However, there are 20 people in the data set. It is not possible to distinguish all these people in a single 2D space. Therefore, we choose subsets for evaluation. Particularly, we choose the first three persons (an2i, at33, boland) and the second three persons (bpm, ch4f, and cheyer). For space limitations, we only show the results from faces of bpm, ch4f and cheyer, since there are more differences in this subset. The result is depicted in Figure 6.9. Each point represents a face image, and the points with same color represent images of the same person. Besides the points, we show some interesting images with different head positions. Clearly, PIM outperforms all the other methods giving the best arrangement. Faces of different people are well separated in the projected space. With 200 samples PIM provide nearly the same



(a) PIM($S = 1k$). (b) PIM($S = 200$).

(c) Hausdorff. (d) SMD.

(e) Netflow. (f) QD($\phi = 0.1$).

Figure 6.9: MDS on subset of Face (bpm, ch4f, and cheyer). Each point represents a face image of one of these three persons. Some interesting images are displayed. Red points: images of bpm, green: ch4f, blue: cheyer.

results as that with 1000 samples. QD is not able to distinguish these faces. Hausdorff, Netflow and SMD perform better. However, they cannot distinguish all the faces from ch4f (green points) and cheyer (blue points). For example, in Figure 6.9c and 6.9e Hausdorff and Netflow put the straight pose face of ch4f closer to the faces of cheyer. And in Figure 6.9d SMD poses the left pose face of ch4f close to the faces of cheyer.

6.4.2 Efficiency of Indexing with PIM

Setting and Data Sets

We support PIM, Hausdorff and Netflow with the Metric Tree (M-tree) [33] and compare the average K-NN query processing time. Further, we compare PIM with M-tree to QKNN (quantile-based KNN) [118] which is, to the best of our knowledge the only specialized technique for indexing multi-instance data. As a baseline for indexing, we also consider sequential scan for each method.

To guarantee comparability, we implement our technique and all comparison methods in C++. For M-Tree the former routing object and the object which is farthest from it are promoted in the splitting process. After several experiments, the node capacity for M-tree is set to 25 and the block size of R-tree (QKNN) is set to 2048. The parameter ϕ of QKNN is set to 0.5. Experiments have been performed on a workstation with 3.4GHz Intel Core i7 and 32G memory.

We follow the way that is used in [118, 122] to generate the synthetic data sets. Specifically, the number of objects N varies from $20K$ to $100k$. Dimensionality D varies from 5 to 25. The centers follow either *Uniform*, *Normal*, or *Mixed* distribution. Mixed distribution is randomly generated from one of the first two distributions. Centers are normalized to $[0, 1]$ in each dimension. The number of instances of an object varies from 500 to $2.5k$. Locations of instances in an object follow either *Uniform*, *Normal*, or *Mixed* distribution. For *Uniform* distribution, instances are located in $[C_i - R, C_i + R]$, where C_i is the center of dimension i and R varies from $[0.05, 0.25]$. The standard deviation of *normal* distribution is set to $0.2 * R$. The instance of Mixed distribution is randomly generated from one of

Table 6.2: Parameters for Synthetic Data Sets

dimensionality D	**5**, 10, 15, 20, 25
number of objects N	**20K**, 40K, 60K, 80K, 100K
edge length R	**0.05**, 0.1, 0.15, 0.2, 0.25
number of instances M	**500**, 1k, 1.5k, 2k, 2.5K
K	5, **10**, 15, 20, 25
object location	Uniform, Normal, **Mixed**
instance location	Uniform, Normal, **Mixed**

the first two distributions. All instances of an object share the same weight. Table 6.2 summarizes the parameters used in our experiments with the default values in **bold** font. In the following, we use default values unless otherwise specified.

For real data sets, we generate two data sets according to [122] from the Forest CoverType (FC) and Household (HOUSE) data sets. Specifically, we select the normalized horizontal distances of each observation point to the Hydrology and roadways as the centers of objects in FC. In House, each record represents the percentage of an American family's annual income spent on 3 types of expenditures. Then instances are generated following the mixed distribution against each center. In FC, there are 581,012 objects and 116.6m instances, while in HOUSE the number is 127,932 and 51.2m respectively.

In all our efficiency experiments, we randomly choose 100 objects as query objects and report the average query time.

Metric Property of PIM

The calculation of PIM involves the process of Monte Carlo integration. The number of samples may affect the metric property of PIM. Therefore, before using M-tree to evaluate the efficiency of PIM, we firstly experiment the effect of S on metric property of PIM. S only affects the accuracy of Monte Carlo integration, thus intuitively enough samples will fulfill the metric property.

We use synthetic data sets with different dimensionality shown in Table 6.2 for evaluation. Specifically, we randomly choose 100 objects from the data sets, and check all the triplets on whether the PIM distance between them fulfill the triangle inequality equation.

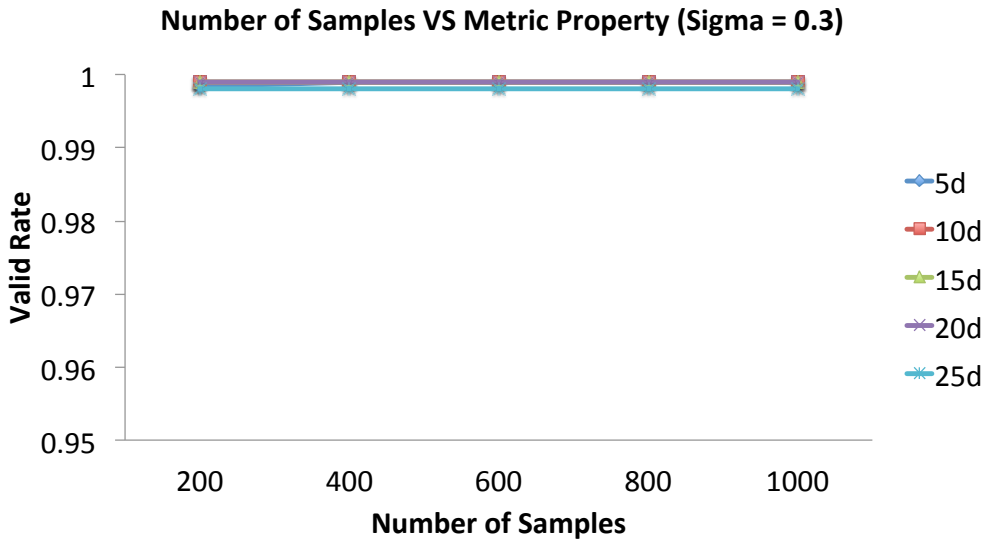


Figure 6.10: Number of samples vs Metric property.

We report the valid rate of those ones that satisfy the equation. From Section 6.4.1 we know that, PIM performs best with $\sigma = 0.3$. Therefore, we evaluate PIM varying number of samples $S = [200, ..., 1k]$. The results are depicted in Figure 6.10. PIM already hold the metric property when number of samples are equal or larger than 200 for all the data sets. For data sets with equal or less than 20 dimensions, 99.9% of triplets fulfill the triangle inequality equation. For $25d$ data, the percentage is 99.8%. In the following, we will evaluate indexing with PIM using samples $S = 1k$ and $S = 200$.

Efficiency

We evaluate the efficiency of PIM by comparing to competitors on performing indexing against synthetic and real data sets. The index construction time of M-Tree with Netflow is longer than 1 week even for the smallest data set. Thus it is eliminated from our efficiency experiments.

We firstly study the scalability of PIM with M-Tree indexing on synthetic data sets varying the dimensionality (D), number of instances (M), objects size (N), the K in K-NN query processing and the edge length (R), see Table 6.2. The average query time is shown in Figure 6.11. To better distinguish the performances between PIM and QKNN, we

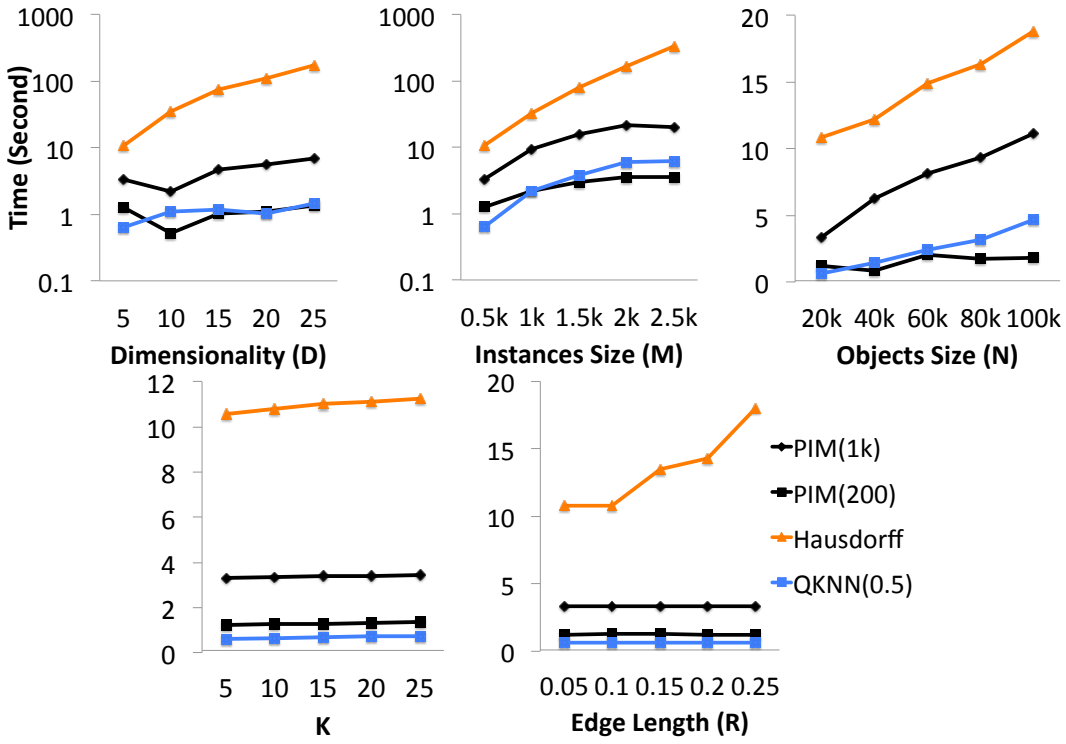


Figure 6.11: Scalability experiments on synthetic data sets in Table 6.2.

use logarithmic scale for D and M and normal scale for the others. PIM and QKNN scale linearly regarding dimensionality (D), while Hausdorff scale super-linearly. The calculation of Hausdorff is linear in terms of dimensionality. Thus this is mainly caused by the bad index selectivity of Hausdorff. In terms of instances size, Hausdorff also scales super-linearly, while PIM and QKNN are linear. This is because the calculation of Hausdorff is quadratic with instances size, see Section 6.3. With more instances (M $>$ 1.5k), PIM with sample size $S = 200$ performs better than QKNN. Regarding objects size N and K in K-NN, all methods scale linearly, where PIM with $S = 200$ performs best with more objects (N $>$ 60k). Edge length (R) indicates the overlapping of instances between objects. PIM and QKNN are not affected by R, while the query time with Hausdorff significantly increased with R $>$ 0.15. Overall, PIM and QKNN give similar scalability in our experiments. With sample size $S = 200$, PIM provides better query response time than QKNN when there are more instances and objects. Due to the bad index selectivity, Hausdorff performs worst in all the experiments.

As real data we take the Forest Covertype and HOUSE data sets. The results are shown in Figure 6.12 and 6.13.

The average processing time for 10 nearest neighbor query on Forest Covertype data set is shown in Figure 6.12. PIM profits very much from indexing: With M-tree support, PIM with $S = 1k$ is around 489 times faster than without index support. For sample size $S = 200$ indexing still yields a speedup factor of about 465. The superior index selectivity demonstrates the effectiveness of PIM. Compared to PIM, Hausdorff profits by indexing with a speedup factor of around 339 in average query processing time. The worse index selectivity of Hausdorff indicates its less effectiveness compared to PIM. QKNN achieves a even worse speedup factor of 196 and it provides a higher response time of 5.71 seconds. In comparison, PIM with sample size 200 provides the second fastest average query response time of only 1.49 seconds. Overall, PIM with M-tree provides the best speedup factor and very fast query time with proper samples on Forest Covertype data.

Figure 6.13 depicts the results on HOUSE data set. Similar to that on Forest Covertype data, PIM profits from indexing supported by M-tree. PIM with $S = 1k$ is about 153 times faster than sequential scan. For sample size $S = 200$ indexing still provides a speedup factor of about 132. Hausdorff does not profit that much by indexing with a speedup factor of only 60. QKNN achieves the best speedup factor of around 359 on HOUSE data set. Regarding response time, PIM with $S = 200$ is the featest among them all.

From Section 6.4.1, we know that PIM with $S = 200$ is only slightly worse than $S = 1k$ and much better than Hausdorff and QD in terms of effectiveness. Therefore, PIM with $S = 200$ is a better method than Hausdorff and QKNN in both effectiveness and efficiency. Even with $S = 1k$ samples, PIM is much efficient than Hausdorff. Moreover, PIM with M-tree provides higher speedup factor than QKNN in FC data, but lower one in HOUSE data. This is mainly because there is cluster structure inside FC data. This indicates that PIM with M-tree performs better on data sets with cluster structure than QKNN. Besides, note that QKNN is a specialized technique for efficient K-NN query processing on multi-instance data and not just a basic M-tree. QKNN is specially designed for K-NN query processing and cannot perform range queries, while PIM with M-Tree can do both

K-NN and range queries directly inherited from the M-tree.

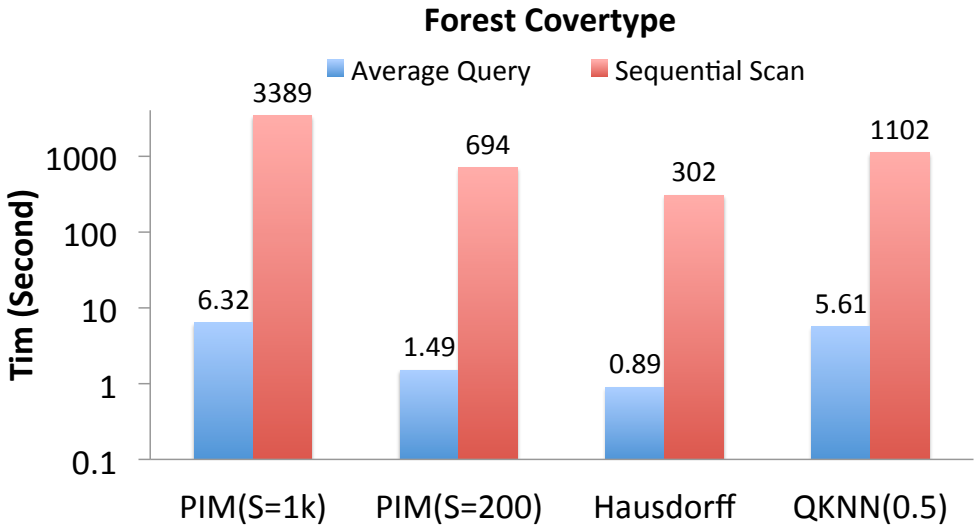


Figure 6.12: Average query processing time on Forest Covertype data.

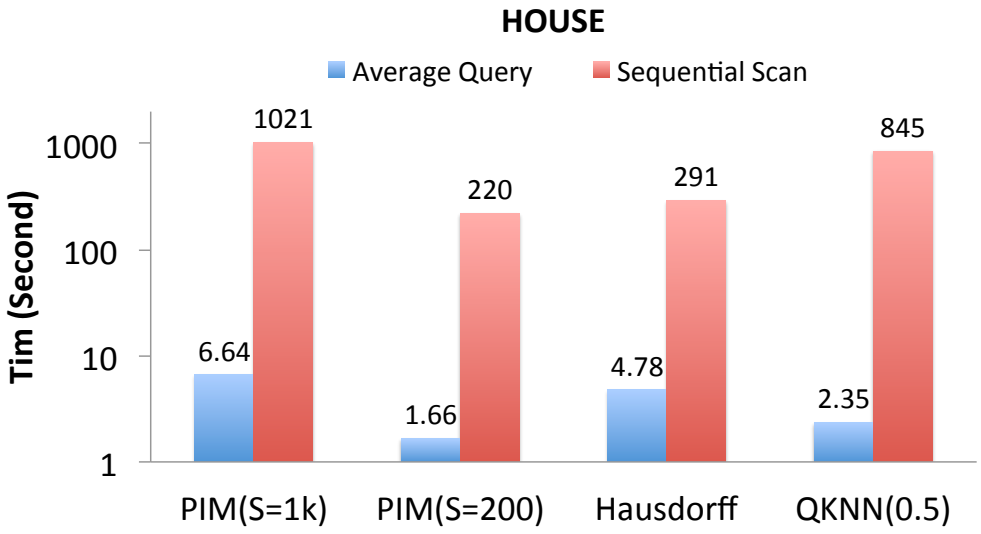


Figure 6.13: Average query processing time on HOUSE data.

6.5 Related Work and Discussion

This section gives a survey and discussion of similarity measures and indexing techniques for multi-instance data.

6.5.1 Similarities for Multi-instance Data

Recently, various similarity measures for multi-instance data have been proposed, ranging from relatively simple and efficient proposals like the Hausdorff distance and the Sum of Minimum Distances (SMD) to more sophisticated measures like the Quantile-distance (QD) and Netflow distance [92, 41, 118, 96]. These similarity measures suffer from one or several of the following drawbacks: (1) Many of the mentioned similarity measures do not fulfill the metric properties, e.g. SMD and QD, many data analysis and data mining methods are not applicable or cannot be guaranteed to give reasonable results. Moreover, due to the lack of metric properties, these methods cannot be supported by metric index structures. (2) others do not consider all available information, like Hausdorff and QD or (3) are not scalable to large data sets, like the Netflow distance. Let us discuss in more detail the performance of the comparison similarity measures in the experiments.

SMD performs quite reasonable in terms of effectiveness. However, since SMD is not a metric, many data analysis and data mining methods are not applicable or cannot be guaranteed to give reasonable results. For example, classical multi-dimensional scaling must not perform well on non-metric data and in this experiment, the performance of SMD is rather weak, see Figure 6.9d. Moreover, due to the lack of metric properties, SMD cannot be supported by metric index structure.

The Hausdorff distance is a metric but is largely affected by single outlying instances, since actually these instances determine the overall similarity among two multi-instance objects, since only the worst matching instance pair is considered. As expected, Hausdorff performs bad in all effectiveness experiments, e.g. it fails in measuring the NBA team diversity see Figure 6.7, and is the second worst method for the Face data, see Figure 6.8. Hausdorff can be supported by a metric index structure. However, our experiments with the M-tree [33] demonstrate that the weakness of Hausdorff in effectiveness yields a relatively bad index selectivity. Therefore, the performance gains of indexing over sequential scan are relatively low compared to PIM.

With proper instance weighting, the Netflow distance is a metric, i.e. when we set

the weights of all instances within an object such that they sum up to one, and performs relatively well in effectiveness. The effectiveness results are as good as with PIM, with exception of NBA team diversity, where PIM is the only method giving the intuitively correct ranking of teams. Netflow evaluates the multi-instance similarity by defining a transport network between two multi-instance objects. Netflow then transforms the distances of one object to those of the other and the flow required for this transformation corresponds to the final distance. This procedure is cubic in the number of instances, thus Netflow is the least efficient comparison method. Index support is in principle possible but not practical, since the time for index construction is excessive, see Section 6.4.2.

Finally, we also compared to the Quantile-distance (QD) which is used in a specialized technique for efficient K-NN query processing on multi-instance data [118]. The Quantile distance is not a metric and performs worst in nearly all effectiveness experiments, e.g. on similarity query on the Face data. Another example is that QD judges other players more similar to Michael Jordan than Jordan himself, see Table 6.1. With QD the QKNN is very efficient for K-NN query. However, in practice we also require high quality of effective query results.

To summarize, our novel similarity metric PIM achieves everything we want: It is just as effective as the most expensive Netflow distance, on some data even better. Due to its metric properties, PIM can be significantly speed up using metric index structures. With an M-tree, we can speed up the query processing time by a factor about 450 over sequential scan on the Forest Covertype data data which indicates superior index selectivity.

6.5.2 Indexing Multi-instance Data

To the best of our knowledge, only one technique has been proposed for general indexing multi-instance data [118, 119]. In this work, Zhang et al. propose a quantile-based method for efficiently processing K-nearest neighbor queries. The authors use quantiles because they want to capture the data distribution within the instance space. Based on this idea, they use the non-metric Quantile Distance discussed above and propose an efficient

algorithm for K-NN query processing using this distance. The instances of each multi-instance object are stored in a local aR-tree [118] and minimum bounding boxes containing the instances of the multi-instance objects are organized in an R-tree [57]. Sophisticated pruning routines speed up the K-NN search in a seeding and refinement phase. We compare this approach to PIM and the other similarity measures supported by the basic M-tree [33]. The M-tree is based on the idea to exploit the triangle inequality for pruning using routing objects. In our experiments QKNN scales similarly as PIM. Generally, it provides faster query than PIM with $S = 1k$ and slower one than PIM with $S = 200$ with larger number of objects and instances. Moreover, PIM performs better than QKNN on data sets with cluster structure, e.g. Forest Covertype data. Besides, QKNN is specially designed for K-NN query processing and cannot perform range queries, while PIM with M-Tree can do both K-NN query and range queries directly inherited from the M-tree.

Kriegel et al. [71] used the Netflow distance with a different weighting function for similarity search on CAD multi-instance objects. However, it is specifically designed for the cover sequence model [66] for querying 3D CAD objects. The number of instance fo each object is fixed to 7 in [71]. Thus, it makes no sense to use it for indexing general multi-instance objects with different dimensionality and number of instances. Furthermore, our experiments demonstrate that Netflow distance is very computational expensive and cannot be applied for indexing multi-instance data with large number of instances.

6.5.3 Approaches for Uncertain Data

Approaches for uncertain data may seem at first glance related to our approach as they often involve probabilistic models like Gaussian [24] or Graphical Models [36] (for a survey see [110]). However, the characteristics of uncertain data are largely different to those of multi-instance data. The major challenge of query processing is to trade off the score, i.e. the closeness to the query, with the existence probability of a potential result. Existing approaches to query processing on uncertain data can be classified into two methods (1) returning only results that can co-exist in a possible world and (2) methods returning

results that exist in all possible worlds with a high probability [51]. In our setting of multi-instance data, we face a different challenge: how to quantify the similarity among objects which are represented by multiple co-existing instances. When modeling multi-instance objects as uncertain data, we would need to aggregate the distribution of the instances to the corresponding uncertainty model. Since all instances in our setting are certainly existing, we would loose important information. Therefore, we consider in the experiments only approaches focusing on multi-instance data.

6.5.4 Multi-instance and Metric Learning

The other related topic is multi-instance learning [39, 12], where an object is labeled negative if all the instances are negative, and is labeled positive if at least one of the instances is positive. In Multi-instance learning, only specific instances are important for classification and the proportion of these important instances to the total number of instances in a set might be low [112]. Therefore, the aim of Multi-instance learning is quite different with the problem studied in this paper, where considering all available information is essential. Furthermore, one of the most important application of PIM is indexing Multi-instance object, where no Multi-instance learning approaches concern.

Metric or distance learning approaches for multi-instance objects try to learn a metric or distance from a labeled data [112, 115]. Their aim is to learn the distance that distinguishes the objects from different classes most. Therefore, these methods are only applicable in a supervised condition. In comparison, PIM is a similarity metric that benefits both supervised and unsupervised applications, e.g. indexing multi-instance data.

6.6 Conclusion

In this chapter, we introduced the Probabilistic Integral Metric (PIM), a novel similarity measure for multi-instance data. The definition of PIM is based on a probabilistic generative model requiring few assumptions which hold in many applications: We assume that

the concrete multi-instance objects within a database have been generated by some template multi-instance objects which represent characteristic patterns of the data distribution in the instance space. Note that this generative model does not imply any distribution assumption in the instance space but only assumes that each concrete instance has been sampled from a template instance with some error variance.

Our extensive experiments on synthetic and real data demonstrate the effectiveness of PIM for similarity search, exploratory data analysis and data mining. PIM outperforms established similarity measures in effectiveness because the generative model includes all available information into the distance calculation, i.e. all instances of a multi-instance object and their spatial location. PIM considers two multi-instance objects as similar if they have been generated by a common template, which means that most instances of one object have matching instances in the other object. However, due to the probabilistic generative model, not all instances must have matching partners. Therefore, PIM not only measures the similarity among two multi-instance objects very accurately but also is very robust against outlying instances. Efficiency experiments demonstrate that PIM scales better than other similarity measures for multi-instance data, like the Hausdorff and Netflow distance. Moreover, since PIM is a metric, many index structures are available to speed up similarity search. We demonstrated that PIM supported by an M-tree can even outperform a specialized technique for K-nearest neighbor query processing on multi-instance data especially on large data represented by massive amounts of instances.

Chapter 7

Conclusion and Future Work

This thesis has been focusing on proposing novel clustering algorithms that solve the challenges in mining complex high-dimensional data by integrating different data mining methods. We analyze four different types of data: numerical data, categorical data, bipartite graph data, and multi-instance data. For the first three data resources, we study the relationship between different data mining tasks, e.g. clustering, pattern mining, dimensional reduction, and prediction. Furthermore, we integrate them into novel algorithms. For multi-instance data, we provide a novel similarity measure that facilitates similarity search, clustering and classification on multi-instance data. Our results show that clustering can be improved by combining with other data mining tasks and novel patterns can be discovered from the complex high-dimensional data. In the following, we conclude the major contributions of this thesis and point out the possible future directions.

7.1 Parameter-free Relevant Subspace Clustering

In real applications, clusters usually exist in the subspace of the original feature space. Subspace clustering approaches suffer from the problem of producing redundant results and depending on parameters. ROCAT is designed to unveil truly relevant overlapping subspace clusters in categorical data. It automatically detects most informative subspace clusters by

integrating clustering, feature selection and pattern mining without any input parameter in an information theoretic way. Relating clustering to data compression, ROCAT does not require users to choose a similarity measure to quantify the pair-wise similarity among categorical data objects. The resulting subspace clusters might be overlapping in both objects and attributes set. The relevance of each cluster is validated by its contribution to compress the data. In such way, ROCAT naturally avoids undesired redundancy in clusters and subspaces by allowing overlap only if it improves the compression rate. Besides, ROCAT is robust to noisy objects and noisy attributes which are flexibly identified during the clustering process. Furthermore, ROCAT satisfies the efficiency requirement for big data which scales linearly in data size and quadratic in dimensionality. A preliminary version of this work has been published in [60], where Xiao He made the main contribution.

ROCAT studies the problem of automatically finding the most relevant and non-redundant patterns given a high-dimensional data set without any primary knowledge. It focuses on categorical data, which is definitely necessary to extend it for numerical data or mixed-type data. For numerical data, the Gaussian distribution assumption could be used to compress the data. However, how to find the initial clusters would be more difficult in such case. Moreover, it is very interesting to extend this idea to multi-component data, e.g. time-series data and multi-instance data. Time-series data could be compact in a period of time and instances of a multi-instance object might be from different distributions. Detecting such relevant subspace clusters without input parameters is potentially beneficial for the following data analysis steps. Moreover, it is interesting to apply the algorithms to very large data sets, e.g. gene expression data analysis and recommendation systems.

7.2 Hierarchical Visualization for Subspace Clusters

Another tackled challenge for clustering high-dimensional data is how to interpret the results. Meaningful clusters often exist in different arbitrarily-oriented subspaces of the original feature space. While many techniques exist for detecting them, only very few

approaches touch the challenge of interpreting them. MSS is proposed as a hierarchical subspace clustering algorithm that integrates the supervised dimensional reduction method (Orthogonal Linear Discriminant Analysis) and clustering methods (K-means), as well as Kernel Density Estimation technique. It finds multiple low-dimensional subspaces with correlated feature vectors, each exhibiting an interesting cluster structure. The hierarchical visualization is provided with different low-dimensional subspaces for an intuitive interpretation of the clustering result. With the hierarchical visualization, MSS improves the analysis of clustering results by providing the information of relationship between clusters. Besides, MSS avoids specifying the dimensionality of subspace that is hard to estimate and MSS is robust to irrelevant dimensions. Parts of the material presented in this work have been submitted in [61], where Xiao He made the main contribution.

MSS partitions the data set and provides hierarchical visualization in different levels. It is interesting to extend the visualization idea for non-partition clustering tasks, e.g. Multi-view Clustering. Besides, MSS adopts the idea of integrating supervised techniques into clustering. Extending the work for other supervised learning and clustering methods is another possible direction, e.g. using non-linear Linear Discriminant Analysis for non-linear correlation clusters, integrating with DBSCAN for density-based subspace clusters. Moreover, the efficiency would be a problem while applying MSS to very high-dimensional data, since MSS needs an iterative adaptation of LDA. Using incremental Linear Discriminant Analysis could be one possible solution to solve the problem. Furthermore, it is interesting to apply the algorithm to large real applications, e.g. image segmentation.

7.3 Summarization-based Co-clustering

SCMiner focuses on the relational bipartite graph data. It reduces a large bipartite input graph to a highly compact representation which integrates co-clustering, summarization, link prediction and the discovery of the hidden structure of a bipartite graph. The principle of data compression also known as the Minimum Description Length Principle (MDL) [97] is the basis of SCMiner. Controlled by the MDL principle, SCMiner discovers the

major clusters of both vertex types as well as the major connection patterns between those clusters. In addition, SCMiner predicts missing or future links and removes noisy edges. Based on data compression, SCMiner does not rely on any input parameters which are difficult to estimate. Parts of the material presented in this work have been published in [45], where Xiao He contributed for the development of the main concept, implemented the main algorithms and wrote part of the paper.

One possible direction of extending the idea of summarization-based mining would be to consider the *weighted* bipartite graphs and to apply it to unsupervised user rating data. It is a challenge to summarize the weighted graph, since the weights do not follow the usual distribution assumptions. It costs a lot of bits to encode the weight difference between each vertex and its representing vertex. One possible solution would be using lossy compression and adopting rate-distortion principle for model selection. Besides, many real data can be modeled by multi-relation graph, e.g. K-partite graph. It is interesting to see the effects of applying summarization idea on K-partite graph. Furthermore, one issue of SCMiner is that it only discovers the locally dense connected clusters and fails to detect global communities that are normally sparse. Finally, it is interesting to study the problem of summarizing a sparse graph.

7.4 Mining Multi-instance Data

In application domains like life sciences, multimedia retrieval, and statistical analysis, data is often modeled as multi-instance objects. It is important to measure the similarity between such kind of data objects. Thus we propose a novel similarity measure PIM (Probabilistic Integral Metric) for multi-instance data. The definition of PIM is based on a probabilistic generative model requiring few assumptions. We assume that the concrete multi-instance objects within a database have been generated by some template multi-instance objects which represent characteristic patterns of the data distribution in the instance space. Note that this generative model does not imply any distribution assumption in the instance space but only assumes that each concrete instance has been sampled from a

template instance with some error variance. PIM is a metric which is essential for effective data mining and efficient similarity search. Experiments demonstrate that PIM performs and scales better than other similarity measures for multi-instance data. Besides, since PIM is a metric, many index structures are available to speed up similarity search. Parts of the material presented in this work have been submitted in [62], where Xiao He made the main contribution to this work.

One interesting possible direction of future work would be extending the PIM for Subspace Clustering on Multi-instance data. A subset of instances might follow the generative model while the whole set does not. Mining the most informative subspace clusters for Multi-instance is interesting as well. Besides, extending PIM for efficient indexing structure is another possible direction. PIM is perfect for metric tree, but a specific indexing structure for Multi-instance data is essential to further improve the efficiency.

Bibliography

[1] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Finding hierarchies of subspace clusters. In *PKDD*, pages 446–453, 2006.

[2] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek. Detection and visualization of subspace cluster hierarchies. In *DASFAA*, pages 152–163, 2007.

[3] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. On exploring complex relationships of correlation clusters. In *SSDBM*, page 7, 2007.

[4] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek. Robust, complete, and efficient correlation clustering. In *SDM*, 2007.

[5] E. Achtert, C. Böhm, P. Kröger, and A. Zimek. Mining hierarchies of correlation clusters. In *SSDBM*, pages 119–128, 2006.

[6] E. Achtert, H.-P. Kriegel, and A. Zimek. Elki: A software system for evaluation of subspace clustering algorithms. In *SSDBM*, pages 580–585, 2008.

[7] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD Conference*, pages 61–72, 1999.

[8] C. C. Aggarwal and C. K. Reddy, editors. *Data Clustering: Algorithms and Applications*. CRC Press, 2014.

[9] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD Conference*, pages 70–81, 2000.

[10] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD Conference*, pages 94–105, 1998.

[11] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[12] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, pages 561–568, 2002.

[13] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. Limbo: Scalable clustering of categorical data. In *EDBT*, pages 123–146, 2004.

[14] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *SIGMOD Conference*, pages 49–60, 1999.

[15] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney. Model-based overlapping clustering. In *KDD*, pages 532–537, 2005.

[16] D. Barbará, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *CIKM*, pages 582–589, 2002.

[17] A. Ben-Dor, B. Chor, R. M. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB*, pages 49–57, 2002.

[18] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[19] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The x-tree : An index structure for high-dimensional data. In *VLDB*, pages 28–39, 1996.

[20] P. Berkhin. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, pages 25–71. 2006.

[21] C. Böhm, C. Faloutsos, J.-Y. Pan, and C. Plant. Robust information-theoretic clustering. In *KDD*, pages 65–75, 2006.

[22] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *ICDM*, pages 27–34, 2004.

[23] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *SIGMOD Conference*, pages 455–466, 2004.

[24] C. Böhm, A. Pryakhin, and M. Schubert. The gauss-tree: Efficient object identification in databases of probabilistic feature vectors. In *ICDE*, page 9, 2006.

[25] D. Boley and D. Cao. Training support vector machines using adaptive clustering. In *SDM*, pages 126–137, 2004.

[26] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In *SDM*, pages 243–254, 2008.

[27] S. Brin. Near neighbor search in large metric spaces. In *VLDB*, pages 574–584, 1995.

[28] E. Cesario, G. Manco, and R. Ortale. Top-down parameter-free clustering of high-dimensional categorical data. *IEEE Trans. Knowl. Data Eng.*, 19(12):1607–1624, 2007.

[29] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.

[30] C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD*, pages 84–93, 1999.

[31] Y. Cheng and G. M. Church. Biclustering of expression data. In *ISMB*, pages 93–103, 2000.

[32] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SDM*, 2004.

[33] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[34] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[36] A. Deshpande, L. Getoor, and P. Sen. Graphical models for uncertain data. In C. Aggarwal, editor, *Managing and Mining Uncertain Data*. Springer, 2009.

[37] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[38] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.

[39] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.*, 89(1-2):31–71, 1997.

[40] C. H. Q. Ding and T. Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *ICML*, pages 521–528, 2007.

[41] T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Inf.*, 34(2):109–133, 1997.

[42] 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.

[43] C. Faloutsos and K.-I. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *SIGMOD*, pages 163–174, 1995.

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

[45] J. Feng, X. He, B. Konte, C. Böhm, and C. Plant. Summarization-based mining bipartite graphs. In *KDD*, pages 1249–1257, 2012.

[46] D. Fradkin. Clustering inside classes improves performance of linear classifiers. In *Tools with Artificial Intelligence, 2008. ICTAI '08. 20th IEEE International Conference on*, volume 2, pages 439–442, 2008.

[47] Q. Fu and A. Banerjee. Bayesian overlapping subspace clustering. In *ICDM*, pages 776–781, 2009.

[48] G. Gan and J. Wu. Subspace clustering for high dimensional categorical data. *SIGKDD Explorations*, 6(2):87–94, 2004.

[49] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus - clustering categorical data using summaries. In *KDD*, pages 73–83, 1999.

[50] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[51] T. Ge, S. Zdonik, and S. Madden. Top-k queries on uncertain data: on score distribution and typical answers. pages 375–388. ACM, 2009.

[52] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Discovery Science*, pages 278–289, 2004.

[53] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625–628, 2005.

[54] P. D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[55] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *ICDE*, pages 512–521, 1999.

[56] S. Günnemann, I. Färber, K. Virochsiri, and T. Seidl. Subspace correlation clustering: finding locally correlated dimensions in subspace projections of the data. In *KDD*, pages 352–360, 2012.

[57] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[58] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[59] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *Proc. of SDM Workshop on Link Analysis*, 2006.

[60] X. He, J. Feng, B. Konte, S. T.Mai, and C. Plant. Relevant overlapping subspace clusters on categorical data. In *KDD*, page in press, 2014.

[61] X. He, S. Goebl, S. T.Mai, C. Böhm, and C. Plant. Multiple subspace selection for hierarchical clustering and visualization. Submitted for publication.

[62] X. He, L. Zhou, J. Feng, C. Plant, and C. Böhm. A probabilistic integral metric for multi-instance objects. Submitted for publication.

[63] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, pages 58–65, 1998.

[64] L. B. Holder, D. J. Cook, and S. Djoko. Substucture discovery in the subdue system. In *KDD Workshop*, pages 169–180, 1994.

[65] Z. Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discoery*, 1997.

[66] H. V. Jagadish. A retrieval technique for similar shapes. In *SIGMOD Conference*, pages 208–217, 1991.

[67] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[68] L. Katz. A new status index derived from sociometric analysis. *PSYCHOMETRIKA*, 18(1):39–43, 1953.

[69] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[70] K.-N. Kontonasios and T. D. Bie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *SDM*, pages 153–164, 2010.

[71] H.-P. Kriegel, S. Brecheisen, P. Kröger, M. Pfeifle, and M. Schubert. Using sets of feature vectors for similarity search on voxelized cad objects. In *SIGMOD Conference*, pages 587–598, 2003.

[72] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1), 2009.

[73] P. Kröger, H.-P. Kriegel, and K. Kailing. Density-connected subspace clustering for high-dimensional data. In *SDM*, pages 246–256, 2004.

[74] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[75] J. Kunegis, E. W. D. Luca, and S. Albayrak. The link prediction problem in bipartite networks. *CoRR*, abs/1006.5367, 2010.

[76] D. Lee and J. Lee. Dynamic dissimilarity measure for support-based clustering. *IEEE Trans. Knowl. Data Eng.*, 22(6):900–905, 2010.

[77] D. Liben-Nowell and J. M. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[78] R. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252, 2010.

[79] B. Long, X. Wu, Z. M. Zhang, and P. S. Yu. Unsupervised learning on k-partite graphs. In *KDD*, pages 317–326, 2006.

[80] B. Long, Z. M. Zhang, and P. S. Yu. Co-clustering by block value decomposition. In *KDD*, pages 635–640, 2005.

[81] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[82] D. Luo, C. H. Q. Ding, and H. Huang. Linear discriminant analysis: New formulations and overfit analysis. In *AAAI*, pages 417–422, 2011.

[83] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, 1967.

[84] S. T. Mai, X. He, N. Hubig, C. Plant, and C. Böhm. Active density-based clustering. In *ICDM*, pages 508–517, 2013.

[85] M. Mampaey, N. Tatti, and J. Vreeken. Tell me what i need to know: succinctly summarizing data with itemsets. In *KDD*, pages 573–581, 2011.

[86] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[87] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *KDD*, pages 533–541, 2008.

[88] E. Müller, I. Assent, S. Günnemann, R. Krieger, and T. Seidl. Relevant subspace clustering: Mining the most interesting non-redundant concepts in high dimensional data. In *ICDM*, pages 377–386, 2009.

[89] H. S. Nagesh, S. Goil, and A. N. Choudhary. Adaptive grids for clustering massive data sets. In *SDM*, pages 1–17, 2001.

[90] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, pages 419–432, 2008.

[91] M. E. J. Newman. Clustering and preferential attachment in growing networks. *PHYS.REV.E*, 64:025102, 2001.

[92] I. Niiniluoto. *Truthlikeness*, volume 185. Springer, 1987.

[93] 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.

[94] A. K. Poernomo and V. Gopalkrishnan. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *KDD*, pages 697–706, 2009.

[95] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *ICDM*, pages 547–556, 2011.

[96] J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Inf.*, 37(10):765–780, 2001.

[97] J. Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.

[98] H. Shan and A. Banerjee. Bayesian co-clustering. In *ICDM*, pages 530–539, 2008.

[99] J. Shao, X. He, C. Böhm, Q. Yang, and C. Plant. Synchronization-inspired partitioning and hierarchical clustering. *IEEE Trans. Knowl. Data Eng.*, 25(4):893–905, 2013.

[100] J. Shao, X. He, Q. Yang, C. Plant, and C. Böhm. Robust synchronization-based graph clustering. In *PAKDD (1)*, pages 249–260, 2013.

[101] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *Comput. J.*, 16(1):30–34, 1973.

[102] B. W. Silverman and P. J. Green. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[103] A. Stolcke and S. M. Omohundro. Hidden markov model induction by bayesian model merging. In *NIPS*, pages 11–18, 1992.

[104] A. Stolcke and S. M. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *ICGI*, pages 106–118, 1994.

[105] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[106] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, pages 567–580, 2008.

[107] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML*, pages 1073–1080, 2009.

[108] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Min. Knowl. Discov.*, 23(1):169–214, 2011.

[109] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *SIGMOD Conference*, pages 394–405, 2002.

[110] Y. Wang, X. Li, X. Li, and Y. Wang. A survey of queries over uncertain data. *Knowl. Inf. Syst.*, 37(3):485–530, 2013.

[111] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[112] A. Woznica and A. Kalousis. Adaptive distances on sets of vectors. In *ICDM*, pages 579–588, 2010.

[113] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan. Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Discov.*, 23(2):215–251, 2011.

[114] T. Xiong, S. Wang, A. Mayers, and E. Monga. A new mca-based divisive hierarchical algorithm for clustering categorical data. In *ICDM*, pages 1058–1063, 2009.

[115] Y. Xu, W. Ping, and A. T. Campbell. Multi-instance metric learning. In *ICDM*, pages 874–883, 2011.

[116] M. J. Zaki, M. Peters, I. Assent, and T. Seidl. Clicks: an effective algorithm for mining subspace clusters in categorical datasets. In *KDD*, pages 736–742, 2005.

[117] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.

[118] W. Zhang, X. Lin, M. A. Cheema, Y. Zhang, and W. Wang. Quantile-based knn over multi-valued objects. In *ICDE*, pages 16–27, 2010.

[119] W. Zhang, L. Zhan, Y. Zhang, M. A. Cheema, and X. Lin. Efficient top-k similarity join processing over multi-valued objects. *World Wide Web*, 17(3):285–309, 2014.

[120] X. Zhang, F. Pan, and W. Wang. Care: Finding local linear correlations in high dimensional data. In *ICDE*, pages 130–139, 2008.

[121] X. Zhang, F. Pan, and W. Wang. Redus: finding reducible subspaces in high dimensional data. In *CIKM*, pages 961–970, 2008.

[122] Y. Zhang, W. Zhang, J. Pei, X. Lin, Q. Lin, and A. Li. Consensus-based ranking of multivalued objects: A generalized borda count approach. *IEEE Trans. Knowl. Data Eng.*, 26(1):83–96, 2014.

Acknowledgments

During the time I was working on this thesis I have been fortunate to receive a lot of supports and encouragements. I would like to express my sincere appreciation to the people who helped me grow academically and personally during my study in Germany.

First, I would like to express my deep gratitude to Prof. Dr. Christian Böhm, my supervisor, for his patient supervision, enthusiastic encouragement and endless support during this research. His research style, sharp thoughts, open-mindedness inspired me all the time. Then I would like to offer my special thanks to Dr. Claudia Plant for her pleasant cooperation, excellent guidance, and friendly encouragement. Furthermore, I am also greatly thankful to Prof. Xingquan Zhu, who kindly agreed to allocate his time to be the second referee on this thesis.

My warmest thanks also to all my current and past colleagues at the data mining group of LMU and iKDD group of Helmholtz Zentrum München. Without the discussions and cooperations with them, this thesis could never have been grown. In particular, I want to thank Dr. Junming Shao, Qinli Yang, Dr. Bianca Wackersreuther, Dr. Annahita Oswald, Jing Feng, Son Mai Thai, Bettina Konte, Sebastian Goebl, Nina Hubig, Sam Maurus, Annika Tonch, Wei Ye, Linfei Zhou, Dr. Wolfgang zu Castell, Dr. David Endesfelder, Can Altinigneli, Frank Fiedler, Peter Wackersreuther and Andrew Zherdin for productive team-work and inspiring discussions. Furthermore, I want to thank Susanne Grienberger, Sandra Mayer and Franz Krojer for their kindly background and technical supports.

I would like to acknowledge China Scholarship Council and University of Munich for providing me the financial support (CSC-LMU joint Scholarship) during my PhD study.

Finally, I like to express my deepest gratitude to my parents, my wife and my daughter for their understanding, patience and endless love. Without them this work would never have seen its conclusion.

Xiao He
Munich, Germany
July, 2014

Curriculum Vitae

Xiao He received the B.Eng. degree in Electronic Information Engineering and M.Eng. degree in Circuit and System from Xidian University , Xi'an, China. He is currently a Phd student at the Institute for Mathematics, Informatics and Statistics, University of Munich, Munich, Germany. His research interests include data mining, machine learning, pattern recognition, image processing, with a focus on clustering, subspace clustering, feature selection, graph mining, and application scenario of computational biology and bioinformatics.

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

He, Xiao

Name, Vorname

München, 02.09.2014

Ort, Datum

He, Xiao

Unterschrift Doktorand/in

Formular 3.2