Efficient Knowledge Discovery in Subspaces of High Dimensional Databases

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Emmanuel Alexander Müller

aus

Athen, Griechenland

Berichter: Universitätsprofessor
Dr. rer. nat. Thomas Seidl

Universitätsprofessorin
Dr. rer. nat./Griechenland Myra Spiliopoulou

Tag der mündlichen Prüfung: 9. Juni 2010

Acknowledgements

There are many people who supported me during my thesis work. I would like to express my thanks to all of them, even if only some of them are mentioned here.

First of all, I would like to thank Prof. Thomas Seidl, my supervisor and first referee. He made this work possible by offering me the opportunity to work on my own choice of scientific challenges as part of his research group. I benefited a lot from the opportunities he provided for all of us and enjoyed the inspiring working atmosphere he created.

I want to extend my thanks to Prof. Myra Spiliopoulou. She did not only agree to act as my second referee but also shared a lot of her knowledge about general scientific work and especially about data mining with me. The feedback I received from her during my thesis helped me a lot.

Most of the solutions in this thesis were developed in a team and I want to thank the people I published with. Especially, I want to mention Ira Assent, Ralph Krieger, Stephan Günnemann and Ines Färber. The cooperation with them had a major influence on this thesis which I do not want to miss. As we contributed as a team in major parts of our daily work, I want to thank my colleagues and all of my students for many fruitful discussions. As there were more than only discussions about scientific topics, they all provided a very friendly and enjoyable atmosphere.

I would also like to express my deep thanks to Daniel Neider who was a big help in writing down this thesis. He helped me a lot by carefully reading parts of the thesis and offering useful hints for polishing my English and the underlying LaTeX code.

I want to thank my parents for their encouragement and their love during all times of my life. Especially, I want to thank Claudia for all your support and love even in the stressful times. Without all of you, it would have been very difficult to focus on my research.

Abstract

In many recent applications such as sensor network analysis, customer segmentation or gene expression analysis tremendous amount of data is gathered. As collecting and storing of data is cheap, users tend to record everything they can. Thus, in today's applications for each object one uses many attributes to provide as much information as possible. However, the valuable knowledge to be learned out of this information is hidden in subsets of the given attributes. Considering any of these subspaces one expands the search space significantly. This poses novel challenges to data mining techniques which aim at extracting this knowledge out of high dimensional databases.

This work has its focus on clustering as one of the main data mining tasks. Clustering is an established technique for grouping objects based on mutual similarity. As traditional clustering approaches are unable to detect clusters hidden in subspaces of high dimensional databases, recent subspace clustering models have been proposed that detect groups of similar objects in any subset of the given attributes. However, as the number of possible subspaces scales exponentially with the number of attributes, development of efficient techniques is crucial for knowledge discovery in subspaces of high dimensional databases.

In this work we propose both novel subspace clustering models aiming at high quality results and efficient processing schemes for these models. We start with novel subspace cluster definitions ensuring the detection of clusters in arbitrary subspaces. We highlight the general challenges of redundancy in recent subspace clustering models and propose novel non-redundant subspace clustering definitions. In this context, our aim is to reduce result sizes to all and only novel knowledge by optimizing the overall subspace clustering result. According to these models not all subspace clusters are valuable for the final result. Based on this general observation we propose efficient processing schemes. Our novel algorithmic solutions overcome efficiency problems caused by exhaustive search of almost all subspace projections and costly database access. We select only the most promising subspace regions for efficient subspace clustering. Overall, our techniques are scalable to large and high dimensional databases providing only few but high quality subspace clusters.

Furthermore, as a general contribution to the community we provide a systematic evaluation study on a broad set of approaches. We show both efficiency and quality characteristics of major paradigms. As major aspect for sustained scientific research we ensure repeatability and comparability for all of our empirical results. Our evaluation framework is available as open source project and provides a basis for future enhancements in this research area. Thus, this thesis provides not only novel methods for efficient cluster and also outlier detection in subspaces of high dimensional data, but it is a fundamental basis for repeatable comparison of recent data mining approaches.

Zusammenfassung

In vielen modernen Anwendungen wie der Analyse von Sensornetzwerken, Kundensegmentierung oder Genexpressionsanalyse werden große Datenmengen gesammelt. Da Datenerfassung und Speicherung billig sind, werden Benutzer häufig dazu verleitet so viel wie möglich zu erfassen. In heutigen Anwendungen werden somit für jedes Objekt viele Attribute verwendet um so viel Information wie möglich bereitzustellen. Dabei ist jedoch das wertvolle Wissen, welches man aus diesen Informationen gewinnen kann, in Teilmengen der gegebenen Attribute versteckt. Betrachtet man solche Teilräume, so erweitert man den Suchraum signifikant. Dies stellt neue Herausforderungen für Data Mining Techniken dar, die als Ziel haben dieses Wissen aus hochdimensionalen Datenbanken zu extrahieren.

Diese Arbeit untersucht Clustering als eine der Hauptaufgaben des Data Mining. Clustering ist eine etablierte Technik zur Gruppierung von Objekten anhand ihrer Ähnlichkeit zueinander. Da jedoch traditionelle Clusteringansätze nicht fähig sind Gruppierungen in Teilräumen von hochdimensionalen Datenbanken zu erkennen, wurden Subspace Clustering Modelle entwickelt. Diese Modelle erkennen Gruppen von ähnlichen Objekten in Teilmengen der gegebenen Attribute. Da jedoch die Anzahl an möglichen Teilräumen exponentiell mit der Attributzahl steigt, ist die Entwicklung effizienter Techniken zur Wissensextraktion in Teilräumen von hochdimensionalen Datenbanken äußerst wichtig.

In dieser Arbeit stellen wir sowohl neue Subspace Clustering Modelle als auch effiziente Methoden für deren Berechnung vor. Wir beginnen mit neuen Subspace Cluster Definitionen, welche die Erkennung von Gruppierungen in beliebigen Teilräumen ermöglichen. Wir beschreiben dabei allgemeine Herausforderungen, die durch die Redundanz in bisherigen Subspace Clustering Modellen bedingt sind und entwickeln neue redundanzfreie Subspace Clustering Definitionen. Unser Ziel ist dabei die Resultatgröße zu reduzieren um durch eine Optimierung der Ergebnismenge nur neues Wissen auszugeben. Durch diese Modellierung sind nicht alle Subspace Cluster für das Resultat von Relevanz. Basierend auf dieser allgemeinen Beobachtung entwickeln wir effiziente Berechnungsmethoden. Unsere neuen Algorithmen überwinden dabei die Effizienzprobleme, die durch den riesigen Suchraum beliebiger Teilraumprojektionen und auch durch die kostenintensiven Datenbankzugriffe bedingt sind. Hierfür wählen wir nur die erfolgversprechendsten Regionen für das Subspace Clustering aus. Insgesamt sind unsere Techniken auf großen hochdimensionalen Datenbanken anwendbar und geben dabei nur wenige aber dafür hochwertige Subspace Cluster aus.

Als allgemeinen Beitrag für die Forschungsgemeinschaft vergleichen wir in einer systematischen Evaluierungsstudie eine große Anzahl an Verfahren. Wir untersuchen sowohl die Effizienz als auch die Qualität der wichtigsten Paradigmen. Für eine nachhaltige Forschung stellen wir sicher, dass sich alle empirischen Untersuchungen auf reproduzierbare und vergleichbare Ergebnisse stützen. Unser Evaluierungsrahmenwerk stellen wir als Open Source Projekt zu Verfügung. Dieses bietet eine Basis für zukünftige Forschung in diesem Bereich. Diese Arbeit stellt somit nicht nur neue Methoden zur effizienten Erkennung von Clustern aber auch Outliern vor, sondern ist auch Grundlage für einen reproduzierbaren Vergleich neuster Data Mining Techniken.

Contents

Chapter 1

Thesis overview

1.1 Introduction

In today's applications, huge amount of data is collected for various analysis tasks. As measurement and storage of data has become cheap due to automated processes, databases storing large amounts of information are ubiquitous. However, such given information does not provide any use to the application if one is not aware of its hidden knowledge. Automatic analysis techniques are essential for the extraction of knowledge out of these databases. As general solution, Knowledge Discovery in Databases has been described as the KDD process [HK01]. The KDD process includes multiple steps ranging from the raw storage of data up to the knowledge that humans can learn out of the hidden patterns (cf. Figure 1.1). As key step "data mining" covers the extraction of patterns out of a given database. There are multiple different purposes for such pattern extraction described in the literature. This thesis has its focus on "cluster" extraction as fundamental knowledge discovery task. In general, a cluster represents a group of objects showing similar information in the database. Using no prior knowledge about the data, clustering methods detect clusters as the aggregated hidden structure of a database. Clustering algorithms are essential for knowledge extraction as clusters represent novel knowledge derived out of the given database.

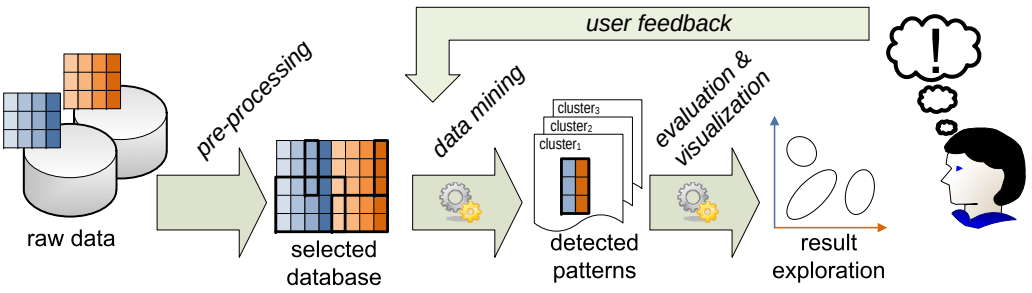


Figure 1.1: KDD process instantiation for subspace clustering

Recent challenges for cluster detection arise out of the data collection and hinder the efficient and accurate detection of the hidden clusters. In most cases data is

collected for multiple analysis tasks, and thus, databases contain objects specified by many attributes. As one does not know the hidden structure of the data, one mixes up all available attributes in one high dimensional database and tries to automatically detect the hidden patterns. However, clusters do not show up on all of these attributes as they are hidden in subsets of the attributes.

Traditional clustering approaches fail on such high dimensional databases as they consider similarity of objects on all given attributes (full space). Due to the so called "curse of dimensionality" [BGRS99], all objects tend to be equally dissimilar to each other in such high dimensional spaces. Thus, no groups of similar objects can be detected. Even after removing some globally irrelevant attributes as pre-processing [Jol86], the curse of dimensionality remains. Each hidden cluster shows up only on a subset of attributes while all of these attributes form a high dimensional space in which neither attributes can be removed, nor clusters can be detected. Overall, traditional methods are unable to detect such clusters hidden in arbitrary subspaces.

Thus, for the general case of high dimensional databases it is essential to provide methods to detect "subspace clusters" as groups of objects showing high similarity only on a subset of the given attributes. Each subspace cluster is described by a set of objects (cluster) and a set of attributes (subspace). Informally, the set of attributes provides the reasons why the objects are grouped together. As there might be multiple reasons why objects group together, one has to search for clusters in all subspace projections considering any subset of attributes. Thus, one is able to detect for each object multiple concepts hidden in the data. These concepts are described by the subspaces which can be seen as different views on the same database. However, searching for multiple concepts per object one might result in many similar clusters. For a high quality subspace clustering, only few of these clusters are relevant while the others are redundant and should be excluded.

In this thesis we focus on the development of novel subspace cluster definitions to formalize these intuitive requirements. As the formal basis for our efficient and high quality solutions, these subspace cluster definitions provide the key properties to be fulfilled by high quality subspace clusters. They characterize properties of the clustered objects in any possible subspace. However, the search in arbitrary subspace projections poses novel challenges not only for the formal subspace cluster definitions but also for their accurate and efficient computation. On the one side, for high quality models we have to tackle novel challenges like redundant subspace clusters. While, on the other side, for an efficient algorithm we have to cope with the computational complexity due to the huge number of possible subspaces in which hidden clusters might occur. And these are only two of the major challenges to be tackled in this work.

In the following we provide a broad overview before discussing our novel subspace clustering methods. We first describe applicability of subspace clustering in real world scenarios in Section 1.2 before introducing the general challenges in Section 1.3. We give a short overview of related work and discuss the general drawbacks of existing approaches in Section 1.4 before describing the main contributions of this thesis in Section 1.5.

1.2 Applicability of subspace clustering

Subspace clustering is applicable to a large variety of applications such as sensor networks, health surveillance, gene expression analysis or customer segmentation. For all of these applications objects are described by many attributes, while groups of objects appear only in subspace projections. As all of the proposed methods in this thesis in general are based on high dimensional data with an underlying Euclidean space, they are applicable on each of these scenarios. We do not provide specialized solutions for any concrete application. We abstract from the individual properties and tackle general challenges observed in a broad set of applications. However, to motivate the challenges of subspace clustering let us consider an intuitive example in one of these application scenarios.

In the area of customer segmentation one is interested in detecting groups of customers that show similar behavior in their shopping activities. Without any prior knowledge about possible groupings, cluster detection is used to detect groupings e.g. for specialized advertisement initiatives. Therefore, the hidden knowledge about customer groups is extracted only based on the gathered customer information such as income, traveling frequency and many more. Each of the customers (objects) is described by these personal properties (attributes) in one high dimensional database. However, as one collects as many attributes as possible (permitted by law or disclosed by willing customers), each customer becomes unique if considering all measured attributes. By using more and more attributes one can distinguish between almost any customer. This raises not only privacy issues for legislation but also leads to low similarity between arbitrary objects in data mining. This observation is an instance of the curse of dimensionality. Although customers show common interests in some of the attributes, there seem to be no clusters in the full data space. Subspace clustering is directly applicable on such high dimensional data, aiming at the detection of groups of customers showing similar behavior using only a subset of attributes.

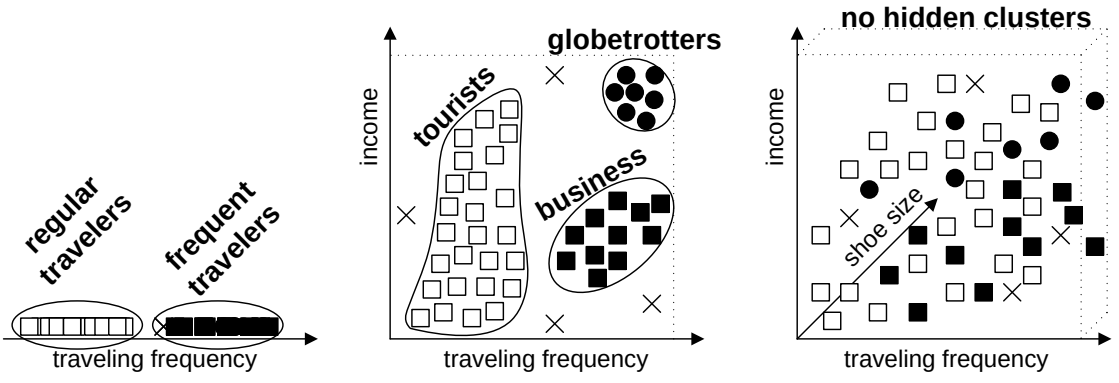


Figure 1.2: Subspace clusters in a customer segmentation example

Let us assume, that there are three different clusters of "traveling types" hidden in our database. For better illustration, we represent these hidden groups by using different symbols in Figure 1.2. As depicted in our small example one might detect the hidden subspace cluster "globetrotters", e.g. showing high income and extremely

high travel frequency. For the distinction between the three types of traveling it is crucial to find the right subspace. The objects are clearly clustered in these two attributes (cf. middle plot), but have almost nothing in common if considering further attributes like "shoe size" (cf. right plot). Including more attributes hinders the detection of clusters as customers diverge from each other and have less in common to be clustered. However, too few attributes do not provide enough information to distinguish between objects (cf. left plot). Considering only the attribute "traveling frequency" does not provide enough information to distinguish between "business travelers" and "globetrotters".

Thus, considering clusters hidden in arbitrary subspaces poses novel challenges for both the formal cluster definition as well as for their efficient computation. Obviously there are efficiency challenges as taking all possible combinations of attributes into account is not practically feasible. While for visualization we have depicted only three subspaces in Figure 1.2, given a database with d attributes there are exponentially many attribute combinations and thus $2^d - 1$ non-empty subsets to be considered.

Further challenges arise for defining clusters in arbitrary subspaces. As illustrated in Figure 1.2 the similarity of objects decreases with increasing number of attributes. For example using the typical Euclidean distance, we observe that distances grow more and more alike. One has to provide a definition which is able to detect the dense "regular travelers" cluster in the 1-dimensional space as well as the scattered "tourists" cluster in the 2-dimensional space.

Assume we have such an adaptive definition, we may compute all hidden subspace clusters as depicted in Figure 1.2. However, do all of these clusters provide novel knowledge? While "frequent travelers" seem to be a generalization of "globetrotters" and "business" travelers, the "regular travelers" and "tourists" are more or less identical sets of objects. Such redundant information should be excluded to provide a small but high quality set of clusters.

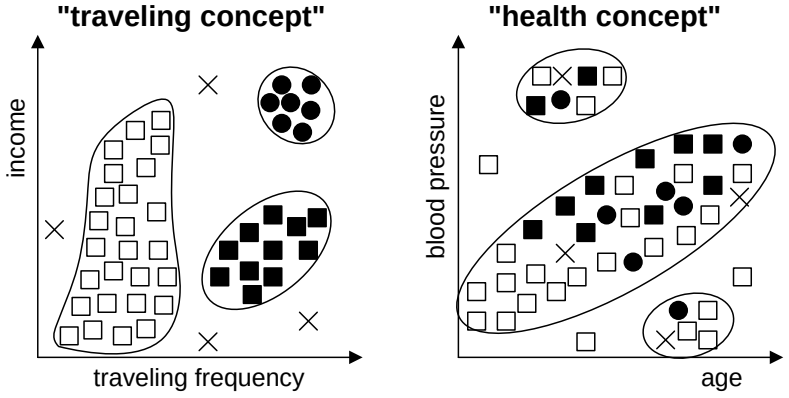


Figure 1.3: An example for multiple relevant concepts

In contrast to redundancy of clusters, different subspaces represent different reasons for grouping objects. Each subspace provides a different view on the same database. Thus, clusters may overlap in their clustered objects in different subspaces, i.e. each object may be represented in multiple subspace clusters. As de-

picted in Figure 1.3, one of the "globetrotters" might be part of a health related cluster of a special disease, while all other "globetrotters" do not have this disease. Furthermore, not only some objects may overlap, but all objects might be clustered in multiple hidden concepts. As depicted in Figure 1.3 we observe some clusters in the "traveling concept" while the same objects form different groupings considering the "health concept". Concepts are described by different sets of attributes. As meaningful clusters appear only in these specific subspace projections of the data, some attributes are useful for the distinction of one concept, other attributes like "age" are irrelevant for grouping in the "traveling concept" but required for clusters like those in the "health concept".

All clusters in these two concepts are valid characterizations of the same objects by using different attributes. Both clusters "tourists" and "high blood pressure" patients should be included in the result. Although they share some objects, they are obviously not redundant. Thus, enhancing our redundancy notion by including multiple concepts is essential for subspace clustering. Furthermore, as shown in our example, some objects might be clustered in one subspace while not included in any cluster in another subspace. This raises additional challenges not only for clustering of noisy data but also for detecting this noise as outliers hidden in subspaces.

1.3 Open challenges of subspace clustering

Abstracting from the mentioned application scenarios there are several open challenges in the area of subspace clustering. Let us summarize these challenges to derive an overview of requirements for our novel subspace clustering techniques. The following chapters then focus on the development of efficient and accurate techniques to tackle each of these challenges.

Challenge 1. Adaptive cluster definition

The first challenge is derived out of the cluster definition itself. While traditional clustering methods consider only one space (full data space) they provide one global cluster definition. Each set of objects which fulfills this definition is a cluster. In contrast, subspace clustering searches for clusters in arbitrary subspaces. Using one cluster definition for all of these subspaces might miss some meaningful clusters. A subspace cluster definition should *adapt* to the considered subspace. As main property of each subspace, the dimensionality has major influence on the data distribution, and thus, it should be utilized to adapt the subspace cluster definition. As distances grow with increasing dimensionality, objects are expected to be dense in low dimensional subspaces while scattered in high dimensional spaces. To detect meaningful clusters in any dimensionality, cluster definition should adapt w.r.t. to this phenomenon.

Challenge 2. Detection of multiple concepts

A subspace clustering as the final set of subspace clusters should allow the detection of multiple clusters for each object. Detecting clusters in arbitrary subspaces

provides a set of attributes which can be seen as the reasons for the grouping of objects. Each subspace represents one of the multiple hidden concepts in a high dimensional database. As observed in several application scenarios, each object might be part of multiple clusters in different subspaces. Assigning each object to at most one cluster would restrict subspace clustering to the detection of a single concept per object. Thus, the result misses other clusters representing additional concepts in other subspaces. A subspace clustering should allow the detection of multiple concepts. We consider overlapping of subspace clusters as a major requirement. Furthermore, in subspace clustering one should actively search for different concepts hidden in the data. The general aim is to detect multiple subspace clusters for all objects each of them representing a different view on the data.

Challenge 3. Redundancy of subspace clusters

In contrast to the benefits of detecting multiple concepts, one has to cope with enormous amount of possible clusters in arbitrary subspaces. In an ideal case, all multiple concepts provide additional knowledge for the overall result set. However, this is not true for redundant subspace clusters. Removal of redundancy is an important challenge for subspace clustering to reduce the result to few but relevant subspace clusters. Redundant clusters should be removed as in contrast to different concepts they provide very similar knowledge to the already detected clusters. There are several redundancy issues to solve, ranging from simple projections of subspace clusters inducing redundant results up to optimization problems of overall result set.

Challenge 4. Computational complexity of subspace clustering

While the previous challenges address the quality of subspace clustering results, one has also to consider an efficient computation. In contrast to traditional clustering approaches which have to show scalability with respect to increasing number of objects in the database, subspace clustering techniques additionally have to scale with the number of given attributes per object. Both are essential for a good runtime performance, and thus, also for the applicability of subspace clustering to large and high dimensional databases. We show that costly database access like in traditional approaches and the exponential search space of arbitrary subspaces pose challenges for an efficient computation. Especially, for our enhanced optimization models we prove that mining the most relevant subspace clusters is NP-hard. This poses novel challenges for the development of approximative but efficient algorithms.

Challenge 5. Outliers hidden in subspaces

As one considers object similarity in arbitrary subspaces, each object might cluster in multiple subspace clusters. However, some objects might cluster in one subspace while showing high deviation from clustered objects in some other subspace. These outlier objects are hidden in subspace projections and deviate only in some of the given attributes. On the one side this noise poses challenges for the definition of subspace clusters. On the other side, outliers can be seen as a different type of desired patterns. As a complementary task to subspace clustering, outlier detection

in subspaces is highly related to the previous challenges. However, these outliers are not simply given as the residual objects (not part of any subspace cluster). Outlier detection in subspaces is a challenging task by its own. Especially, measuring the outlierness of objects in some relevant subspace projections seems to be the most challenging task for outlier detection in subspaces.

Challenge 6. Exploration and evaluation of subspace clusters

As a natural property for clustering, no knowledge is given about the hidden structure of the data. This poses a major challenge to evaluation of subspace clustering results. One possible but quite subjective way of evaluation is visual exploration of results by domain experts. However, for the young research area of subspace clustering exploration tools are not available but strongly desired. A second more objective way of evaluation is the use of labeled data assuming that the given class labels represent the hidden cluster structure. While labeled data are widely used for cluster evaluation, there exists no systematic evaluation of subspace clustering techniques. Especially, due to missing standardized evaluation measures and a missing comparability of different implementations the comparison of subspace clustering techniques is highly challenging. Overall, the empirical results in most of the scientific publications on subspace clustering do not provide any objective and systematic comparison. Different paradigms coexist in the literature without any empirical evaluation of their clustering qualities.

1.4 Related work

In the past few years there have been proposed several approaches for clustering in subspace projections of high dimensional databases. Starting with the first approaches, the development has diverged into *subspace clustering* [AGGR98] and *projected clustering* [AWY+99] approaches introducing two major paradigms. The approaches in both paradigms tackle the general problem of clustering in subspace projections but use very different models. Let us first give a short overview of their main characteristics and highlight why none of the proposed methods tackles all of the mentioned challenges.

Subspace Clustering In general, subspace clustering approaches provide a cluster definition for a set of objects considering similarity of objects in a subset of the given attributes. As introduced by the first subspace clustering approach CLIQUE [AGGR98], each set of objects which fulfills this definition in one of the exponentially many subspaces is output as a cluster result. This ensures the detection of multiple clusters per object as stated in Challenge 2. However, as none of the proposed methods considers redundancy of clusters (cf. Challenge 3), most of the approaches output a tremendous number of subspace clusters. It is quite usual to produce several orders of magnitude more clusters than given objects in the database. Increasing the amount of information with such a processing is clearly not meant by the KDD process. Although multiple concepts are detected in various subspaces by such a

huge result, these meaningful clusters are hidden by various other irrelevant clusters. A data mining step would be required to dig these relevant clusters out of the result set.

As second drawback of subspace clustering approaches they show high runtimes for computing the clustering result (cf. Challenge 4). This is directly related to the huge result set, as they consider arbitrary subspace projections they have to perform cost intensive clustering on many different subspaces. The computation cost depends on the underlying cluster model, but in general, most approaches require cost intensive database scans to perform these clustering steps. For some benchmark data with only thousands of objects and some ten or twenty dimensions one can observe a runtime of several days. This is clearly not efficient and especially, does not scale to large and high dimensional real world applications. Please note, that both the exponential number of possible subspaces as well as the costly database scans in each of these subspaces are the reasons for the observed high runtimes. Both have to be tackled for an efficient subspace clustering algorithm.

All of the proposed subspace clustering approaches are aware of the mentioned efficiency problem. There have been proposed several approaches using pruning techniques based on monotonicity properties of their cluster definition. However, all of these monotonicity properties rely on the fixed cluster properties in all considered subspaces. By using high thresholds for these cluster properties one can prune large parts of the search space which results in a fast computation. However, by setting such high thresholds one clearly misses clusters. Only few approaches consider adaptive clustering models tackling Challenge 1. However, all of these approaches use approximate solutions losing some of the hidden clusters. The aim of an efficient and accurate subspace clustering approach would be to realize high quality results with an efficient algorithm tackling all Challenges 1, 3 and 4.

Projected Clustering For the second paradigm, projected clustering approaches aim at a partitioning of the data such that each object is clustered in at most one projected cluster. This paradigm has been introduced by the first projected clustering approach PROCLUS [AWY+99], which assigns each object to a single cluster in one projection. Due to the partitioning of the data, projected clustering approaches show only few clusters in their result and have neither efficiency nor redundancy problems. The strict partitioning can be regarded as extreme redundancy elimination as projected clustering results in a manageable number of clusters, but is not able to detect overlapping clusters. Thus, projected clustering methods are unable to detect multiple concepts per object. They fail in detecting all hidden concepts in multiple subspace projections as they do not address Challenge 2. Overall the projected clustering approaches have a highly restricted clustering definition. By allowing only disjoint clusters several meaningful clusters are lost. In the mentioned application scenarios this is a general drawback which is not acceptable for clustering in subspace projections.

As discussed in this short overview, compared to subspace clustering one observes benefits but also drawbacks of the projected clustering paradigm. Due to the ma-

jor requirement of multiple clusters per object, we focus on the subspace clustering paradigm. However, we will also discuss the differences to projected clustering competitors in the following chapters and evaluate our results compared to partitioning approaches.

Evaluation of Clustering in Subspace Projections For all of the proposed methods evaluation of results is a major problem. In almost all publications either different quality measures or subjective statements from domain experts are used to show the high quality of achieved results. Both result in incomparability of evaluation results. Even for common measures such as the runtime of an algorithm, the results of different publications are incomparable due to different implementations. However, as only few competitors are evaluated in each publication and no systematic evaluation of a broad set of approaches is available, one is forced to compare the results of different publications to obtain an overall view of subspace clustering approaches. Thus, Challenge 6 is of major importance for thorough evaluation and especially for repeatability of results in scientific publications. Not only for research purposes but also for using this research in application scenarios, evaluation and exploration of the obtained results is a key requirement.

	high dimensional databases	adaptive cluster definition	detection of multiple concepts	only non-redundant clusters	efficient algorithms	evaluation study
traditional clustering	−					+
subspace clustering	+	−	+	−	−	−
projected clustering	+	−	−	+	+	
subspace clustering in this thesis	+	+	+	+	+	+

Figure 1.4: Comparison of major clustering paradigms w.r.t. mentioned challenges [(+) fulfill the requirements (−) still open challenges]

Overall we have summarized the characteristics of recent clustering approaches in Figure 1.4. A thorough comparison of both subspace clustering and projected clustering approaches is given in Chapter 10. Our evaluation study provides a broad overview of approaches and a general discussion of their properties. This might be useful for new researchers to get an overview but also for established researchers for a thorough comparison of different paradigms. In contrast, specialized discussion of related work w.r.t. individual challenges is provided for each of our novel techniques in the respective chapters. In general, one might observe that most of the proposed techniques in the literature have focused on extending traditional cluster definitions to subspaces and on development of enhanced models for clusters in subspace projections. Thus, the state-of-the-art methods have missed to address some specific

challenges in subspace clustering. Especially, the detection of few but relevant concepts in subspace projections has not yet been addressed in subspace clustering. And thus, redundancy as a major challenge has not been addressed for almost a decade of research in subspace clustering.

Outlier Detection in Subspaces Projections Finally, considering outliers as the highly deviating objects in subspace projections, there are some traditional outlier detection methods using the full space [KNT00, EKSX96, HXD03, BKNS00, KSZ08], but only few approaches have been proposed for outlier detection in subspace projections [AY01, KKSZ09, LK05]. In general, the full space approaches are widely used and established while the challenging tasks considering subspace projections are not yet addressed in the literature. Especially, Challenge 5 derived by subspace clustering has to be tackled by outlier detection in subspace projections. Traditional methods (designed for one fixed space) cannot simply be used in arbitrary subspace projections as the deviation measures have to adapt to the considered subspace. None of the proposed subspace outlier methods has addressed this challenge. In addition, most of the proposed subspace outlier detection methods rely on a selection of subspaces in which they measure the deviation of objects. This selection of subspaces is crucial for high quality results and has been addressed in the literature only in some very simple cases like random selection of subspaces.

1.5 Overview of contributions

Let us provide an overview of the major contributions in this thesis before going into details in the following chapters.

Adaptive density-based subspace clustering As first contribution, we provide a subspace cluster definition which adapts to the considered subspace. Focusing on the density-based paradigm proposed by DBSCAN for the full space clustering [EKSX96], we develop an adaptive density which is aware of the decreasing expected density in higher dimensional spaces. In general, our density models automatically adapt to the expected density in each subspace and represent density properties of the hidden clusters better than fixed density definitions.

As basic enhancement we propose in Chapter 2 to adapt the density threshold in the density-based cluster criterion. Intuitively, clusters are defined as dense objects separated by sparse regions [EKSX96]. Dense objects have to exceed a certain density threshold. Traditionally, this threshold is fixed, but as density drops for increasing dimensionality we define a monotonically decreasing threshold function. Our threshold function adapts the cluster definition such that objects have to be denser than expected in the considered subspace. As we show in Chapter 2 this enhances the quality of subspace clustering results. We extend this density definition given only for continuous valued attributes to cope also with heterogeneous subspaces. As we show in Chapter 7 we develop a unified density for both continuous and categorical attributes. While the previous approaches adapt the density threshold and keep

measuring the density in a fixed neighborhood, our most recent approach defines a novel density measure by adapting the neighborhood in which the density is measured. Details about our variable density measure are provided in Chapter 3.

All and only relevant subspace clusters As second contribution we provide a novel clustering definition for the final set of resulting subspace clusters. In general, we propose to consider interestingness and redundancy of subspace clusters and perform an optimization to choose all and only the most relevant subspace clusters as a result. The major enhancement is due to the exclusion of redundant clusters. As we include only few but relevant subspace clusters we improve the clustering quality and in addition gain efficiency improvements for our algorithmic solutions.

As first approach in this area we develop a redundancy definition excluding lower dimensional projections of subspace clusters. This pairwise comparison of two clusters, which would be valid results if one considers only the cluster definition (e.g. for density-based subspace clusters), ensures small and non-redundant clustering results. We propose our non-redundant subspace clustering definition in Chapter 2, and show the improvement of both clustering quality and runtime performance due to in-process removal of redundant clusters as described in Chapter 6. In our second redundancy definition we overcome the drawbacks of pairwise comparison and ensure redundancy-free clustering by a global optimization. A cluster is only included if it contributes novel knowledge w.r.t. all other clusters in the clustering result (cf. Chapter 3). In contrast to our first approach, this optimization yields only quality improvements while we have proven that it is an NP-hard problem. This poses new challenges for the development of efficient approximative solutions (cf. Challenge 4). Based on this optimization we further enhance our clustering model by comparing each cluster only with all other clusters in similar subspaces. Very dissimilar subspaces (orthogonal subspaces) provide novel knowledge as different concepts might be hidden in these subspaces (cf. Challenge 2). We propose our orthogonal subspace clustering model in Chapter 4. It actively includes novel knowledge of orthogonal concepts into the final clustering result.

Efficient computation of relevant subspace clusters As third contribution we develop several novel solutions for efficient computation of our enhanced subspace clustering models. In general, these solutions tackle the high cost for database access, prune the exponential search space of arbitrary subspace projections and propose efficient solutions for optimization of clustering result. Overall, these solutions scale well with increasing number of dimensions, which has shown to be the most challenging task for efficient processing. Evaluated on benchmark data sets these solutions show practical runtimes for the computation of high quality clustering results.

As basic solution to tackle the high cost of database access we propose a filter and refinement architecture for subspace clustering. This basic idea of using a grid approximation as filter and a density-based clustering as a refinement step is described in more details in Chapter 5. It is used in all further developments to ensure efficient access to arbitrary subspace regions. Using grid approximations our techniques can

efficiently check whether a subspace region contains a potential cluster or not. Thus, pruning of irrelevant regions can be performed without costly database access. For further pruning of redundant subspace regions we develop a depth-first processing of the possible subspaces. Using this technique one can exclude large parts of the exponential search space as they contain only redundant clusters. In Chapter 6 we describe how this pruning can be realized to perform an in-process removal of redundant clusters. Furthermore, by using our novel index support we propose to unify density-based subspace clustering and frequent itemset mining for an overall efficient mining of heterogeneous data as described in Chapter 7. We provide a thorough comparison between frequent itemset and subspace clustering as different mining paradigms and derive common properties for frequency on categorical data and density on continuous data. By unifying these two mining paradigms for our heterogeneous subspace clustering model we can provide efficient mining of both categorical and continuous attribute types. This is essential for real world data where typically attributes have various different types.

For the more enhanced subspace clustering models using an optimization of the clustering result we have to cope with an NP-hard problem, as mentioned before. We develop a relaxation of the proposed model as described in Chapter 3. This relaxation can be computed by a greedy processing showing both efficient computation and high quality clustering results. Furthermore, as our relevant subspace clustering excludes most of the exponential search space, we develop a jump processing for subspace clustering. This novel processing schema overcomes the efficiency problems of both traditional breadth-first processing as well as our own depth-first processing methods. Described in Chapter 9 we directly jump to the most promising subspace regions without processing the large number of lower dimensional subspaces. A key requirement for such a jump is a high quality estimation of density in arbitrary subspace regions. We develop an efficient but also high quality estimation (cf. Chapter 8). Using only knowledge from two dimensional histograms our method can estimate density in subspace regions without further database access. This extends our basic filter and refinement architecture by a second filter step with even less computation cost.

Evaluation and Exploration of subspace clusters As final contribution for clustering in subspace projection we perform a systematic evaluation of a broad set of techniques. The underlying evaluation measures ensure an objective comparison of the clustering results based on given class labels in the database. Our novel framework for evaluation and exploration of clustering in subspace projections provides the means for a comparable and especially a repeatable evaluation of the main paradigms.

In our evaluation study we provide a thorough characterization in the main properties of each paradigm and their instantiations in different approaches. We compare different evaluation measures and show their applicability depending on the given knowledge about the hidden clusters. Thorough evaluation shows the drawbacks of early subspace clustering methods producing tremendous amounts of clusters. These results form the fundamental basis for our novel non-redundant subspace clustering.

They highlight the requirements of few but high quality subspace clusters as described in Chapter 10. Overall our results can be repeated by using our open source framework proposed in Chapter 11. It includes several clustering approaches, evaluation measures used in the past decade as well as novel visualization techniques for subspace cluster exploration. Thus, our tool forms the basis for future evaluations and development of novel techniques in the emerging research area of subspace clustering.

Outlier detection in subspace projections As last contribution, we show how our subspace clustering techniques can be used to derive outliers hidden in subspaces. We developed several outlier mining techniques that base on our previous contributions. In general, these techniques enable the ranking of outliers deviating only in some relevant subspaces. These subspaces are important for detection of outliers as well as for providing reasons why these objects are highly deviating.

In our first outlier ranking approach we extract outliers in subspace projections as post-processing to our subspace clustering techniques. As described in Chapter 12, we develop novel scoring functions that take a subspace clustering as an input to rank objects according to their deviation in subspace projections. These scorings derive indicators for outliers out of subspace clustering properties. However, being highly dependent on the clustering result we enhance this first solution to direct outlier ranking without prior subspace clustering. We propose an adaptive outlierness measure considering deviation of objects in subspace projections. Similar to our research in subspace clustering we propose an adaptive density for each subspace. In addition we develop a statistical test for selection of relevant subspaces in which the outlierness of objects is measured. In Chapter 13 and 14 we give more details on both the adaptive outlierness and the statistical selection of subspaces.

The selection of meaningful subspaces is of great importance as these relevant subspaces provide the attributes in which outliers can be distinguished from the residual objects. Thus, these subspaces provide the reasons for being an outlier or not. Based on the selected subspaces and the local neighborhoods of outliers in these subspaces we develop novel descriptive components as witnesses for the outlier properties (cf. Chapter 14). These descriptive components provide additional information about the outliers and assist the user in further knowledge extraction. Overall, our outlier methods yield not only high quality outlier detection but provide additional knowledge about possible outlier reasons.

In the following each of these contributions is described in more details in separate chapters. The chapters are structured in four main parts. The first part introduces our novel subspace clustering models with fundamental definitions for each subspace cluster and the overall selection of a set of relevant subspace clusters. The chapters in this part highlight our contributions for high quality subspace clustering. The second part focuses on efficient computation of our enhanced models. We introduce several pruning techniques to ensure high quality but also efficient subspace clustering. The third part discusses the systematic evaluation of subspace clustering results and provides details about our systematic evaluation. In addition to our research on

subspace clustering, the fourth part provides the extensions to outlier detection in subspace projections. Summing up all of these contributions in the last part of this thesis, we show possible future research questions derived out of our work in this research area.

Part I

Subspace clustering models

Chapter 2

Adaptive subspace cluster definition

In application scenarios with many attributes, clusters are often hidden in subspaces of the data and do not show up in the full dimensional space. For these applications, subspace clustering methods aim at detecting clusters in any subspace projection. However, existing subspace clustering approaches fall prey to an effect we call dimensionality bias. As dimensionality of subspaces varies, approaches which do not take this effect into account fail to separate clusters from noise. Independent of other aspects such as efficiency or redundancy, these methods show major drawbacks in their fixed subspace cluster definitions. Using one fixed density thresholds for all subspaces they can not adapt to the varying density in different dimensionalities. Therefore, in this chapter we focus on an enhancement of density-based subspace clustering methods by providing a novel adaptive subspace cluster definition.

We give a formal definition of dimensionality bias and analyze consequences for subspace clustering. In this chapter we propose *DUSC*, a dimensionality unbiased subspace cluster definition based on statistical foundations. It provides a general technique for adapting density to the expected density of arbitrary subspaces. We use this unbiased density measure as basic technique in most of our subspace clustering approaches to detect density-based subspace clusters. In thorough experiments, we demonstrate that our DUSC approach outperforms existing subspace clustering models in terms of accuracy due to its novel subspace cluster definition.

2.1 Motivation and comparison with related work

Subspace clustering generally aims at detecting clusters in any possible attribute combination. As for traditional full space clustering, there exist different paradigms also for subspace clustering. We focus on density-based subspace clustering. In general, density-based approaches have shown to successfully mine clusters even in the presence of noise [EKSX96]. The idea is to define clusters as dense areas separated by sparsely populated areas. Density of an object is measured either by mere counting of objects or by more complex functions on the number and location of objects in the neighborhood. An object is considered dense if its density is above some threshold.

Density-based clustering has been extended to subspace clustering in previous

works [KKK04]. The definition of density typically is similar to that in full space clustering. In density-based subspace clustering, clusters are dense areas separated by sparsely populated areas as in DBSCAN [EKSX96]. For density-based subspace clustering one simply counts objects in a neighborhood taking only the relevant dimensions of a subspace into account. As in traditional approaches one uses a fixed neighborhood and checks for a fixed minimum number of points as density threshold. The distribution of objects inside the neighborhood is ignored and sensitivity to parameter settings is a challenge for arbitrary subspaces. Overall, one simply uses this traditional paradigm to define a subspace cluster definition for arbitrary subspaces without adapting to the considered subspace.

Ignoring the dimensionality of each subspace has serious consequences for the quality of the result. Density in subspaces of different dimensionalities is not comparable. Existing approaches which do not take this effect into consideration hence check incomparable values against the same threshold. Thus, they fail to separate dense from sparse regions across subspaces of different dimensionalities. Assuming a simple setup of uniformly distributed data, we show that density measures which ignore dimensionality cannot distinguish this pseudo-cluster scenario (with no hidden subspace clusters) from true hidden clusters in all subspaces. As a consequence, dimensionality bias means failing at the very core of density-based subspace clustering. Hence, existing subspace clustering algorithms either lose clusters or detect numerous pseudo-clusters depending on the parameter setting.

Several methods have been proposed in the literature that fall prey to this phenomenon. Approaches such as the well known CLIQUE algorithm discretize the data by grid structures and use monotonicity on density of cells for pruning [AGGR98, NGC01, SZ04]. Grids greatly reduce the computational complexity, yet clusters which spread across several cells might be missed. As first enhancement, MAFIA adapts the grid to a variable width of cells [NGC01]. However as major drawback of CLIQUE and MAFIA, both use fixed density thresholds ignoring the dimensionality of the considered subspace. The recent approach SCHISM [SZ04], extends CLIQUE by using a variable threshold to cope with different dimensionalities, yet relies on heuristics and a grid-based discretization for pruning. Consequently, completeness is lost as in all grid-based approaches. In contrast, the SUBCLU approach is based on the density-based paradigm and uses a density monotonicity property to prune subspaces [KKK04]. As its density computation is based on an ε-neighborhood around each object, it does not rely on a grid discretization. However, by using a fixed density threshold like CLIQUE, dimensionality is ignored. While CLIQUE uses fixed density to specify dense cells, SUBCLU prunes non-dense objects based on a fixed density. Thus, also SUBCLU suffers from dimensionality bias, i.e. clusters cannot be separated from noise across subspaces.

In contrast to these related approaches, we focus on eliminating dimensionality bias. We propose a new density-based subspace clustering approach DUSC (dimensionality unbiased subspace clustering) based on statistical foundations which takes the dimensionality into account. We show that this method eliminates dimensionality bias and leads to an adaptive subspace cluster definition for arbitrary subspaces

of different dimensionalities. To ensure efficient mining of density-based subspace clusters, we derive powerful pruning properties out of our novel model.

Summing up, our contributions include:

- definition and analysis of dimensionality bias and its consequences for subspace clustering
- definition of density based on statistical foundations
- dimensionality unbiased subspace clustering model
- powerful pruning properties for our novel model

2.2 Formalization of density-based subspace clusters

In many applications clusters are hidden in subspaces and cannot be revealed by any cluster analysis that mines all dimensions simultaneously. Subspace clustering automatically focuses to the respectively relevant dimensions. Let us first introduce basic notions for subspaces and subspace clusters.

Let $U = [0, v]$ be a universal domain for all dimensions, $DIM = \{1, \ldots, d\}$ be an index set, and $DB \subseteq U^{DIM}$ a d-dimensional database with n objects. A subspace U^S is the projection of U^{DIM} to the r dimensions specified by the index set $S = \{s_1, \ldots, s_r\} \subseteq DIM$. Analogously, let DB^{DIM} denote the original database and DB^S its projection to the dimensions in S. For ease of notation, we refer to a subspace U^S by its index set S. The definition of density-based subspace clusters extends standard notions in density-based clustering [EKSX96]. Let $\|p - o\|^S$ denote the restriction of norm $\|p - o\| : U^{DIM} \to \mathbb{R}$ for any objects $o, p \in DB$ to the dimensions in subspace S. The area of influence of an object o is the neighborhood in subspace S: $\mathcal{N}_\varepsilon^S(o) = \{p | p \in DB, \|p - o\|^S \leq \varepsilon\}$.

Typically, density of an object o is determined by simply counting the number of objects in a fixed ε-range given by $|\mathcal{N}_\varepsilon^S(o)|$. We generalize this idea by assigning weights to each object contained in $\mathcal{N}_\varepsilon^S(o)$. This generalized density is based on *kernel density estimation* [Sil86]. In statistics, kernel estimators are used to estimate density functions from a set of data objects. The weighting function (kernel) weights the observations in the area of influence according to their distance to the center of the kernel. For a meaningful density estimation, weighting functions are non-negative, integrable and symmetric $W(-x) = W(x)$. Further restrictions as normalization of kernels (required in probability density estimation [Sil86]) are not essential for our weighting as the density is not used as probability measure in our case.

Different kernels correspond to differently shaped curves, resulting in slightly different density assessments. Using a rectangular kernel with equal weights for all objects in the area of influence, results in simple counting of objects which corresponds to the SUBCLU approach. In contrast to such an object counting, a weighted density assessment may assign higher values to closer objects and lower values to objects further away. Thus density is more accurately measured than by mere counting of objects within the neighborhood [HK98, Sil86].

Definition 1. Density Measure
Let $\mathcal{W}$ be a weighting function $\mathcal{W} : \mathbb{R} \to \mathbb{R}_{\geq 0}$. Based on $\mathcal{W}$, a generalized density measure $\varphi(o)$ for an object o in subspace S is defined as:

$$\varphi(o) = \sum_{p \in \mathcal{N}_\varepsilon^S(o)} \mathcal{W}\left(\|o - p\|^S\right)$$

Thus, an object o in subspace S is called *dense* if the weighted distances to objects in its area of influence sum up to more than a given density threshold $\varphi(o) \geq \tau$.

2.2.1 Dimensionality Bias

Subspace clustering methods analyze data spaces of different dimensionalities. Therefore, avoiding an effect which we call dimensionality bias is an important issue. Dimensionality bias refers to a dependency of density on the dimensionality of the subspace: as dimensionality increases, average distances between objects increase and cluster radii grow. This general observation has been formulated as the so called "curse of dimensionality" [BGRS99]. As an instance of the curse of dimensionality, the expected density within the area of influence drops accordingly. Thus, ignoring the dependency of density on the dimensionality of the subspace leads to incomparable density values. Incomparable density values pose the following problem: the high discrepancy in density scales of low dimensional or high dimensional subspaces makes it impossible to find a suitable parameter for a fixed density threshold τ. If on the one hand τ is parametrized such that high dimensional clusters with low expected density are detected then numerous excess pseudo-clusters are generated in low dimensional spaces where expected density is high. On the other hand, a parametrization of τ which separates clusters from noise in low dimensional spaces loses clusters in high dimensional spaces. We assume that τ is fixed as dimensionality dependent thresholds can also be incorporated into the density measure (the same argument holds if one were to vary ε).

To obtain comparable density values, unbiased density measures have to be independent of the dimensionality of the subspace. Statistically speaking, this corresponds to the same expected density value regardless of the dimensionality of the subspace.

Definition 2. Dimensionality Unbiased Density Measure
A density measure φ is dimensionality unbiased if its expected density is the same for any two subspaces S_1 and $S_2 \subseteq DIM$:

$$\forall\ S_1, S_2 : \quad E\left[\varphi^{S_1}\right] = E\left[\varphi^{S_2}\right]$$

We now show how dimensionality bias can be eliminated for any density estimator. As the expected density should be the same for any two subspaces, we normalize density estimators with their expected density. For any density measure φ, the normalized measure $\frac{1}{E[\varphi]}\varphi$ is dimensionality unbiased. With linearity property of the expectation, this is straightforward: $E[\frac{1}{E[\varphi]}\varphi] = \frac{1}{E[\varphi]}E[\varphi] = 1$ for all subspaces. Thus, for any two subspaces, normalizing the density measure by the expected value

of the subspace yields comparable density values for any two subspaces S_1 and S_2. Normalization could be achieved by other means such as subtracting the expected value, but, as we will see later, dividing by the expected value simplifies the choice of density parameters in subspace clustering.

2.2.2 An Unbiased Density Estimator

In this section we use statistical analysis to develop an unbiased density measure for subspace clustering. Based on kernel density estimation we use a kernel function which assigns higher values to closer objects and lower values to objects further away [HK98, Sil86]. The most commonly used are Gauss, Epanechnikov, Bisquare and Triangular kernels. Gauss, however, assigns non-zero values to all objects in the database, even to those at very large distances. It is a poor density estimator in terms of efficiency and effectivity [Sil86]. The Epanechnikov kernel is both an efficient and effective choice, since it is computationally efficient and minimizes the mean integrated squared error [Sil86]. Thus, we use Epanechnikov kernel in the following, but in principle any other kernel could be used as well.

Within an area of influence, the Epanechnikov kernel assigns decreasing weights to objects with increasing distance. For a subspace S, the Epanechnikov kernel function K^S is defined as:

$$K^S(x) = \begin{cases} \frac{|S|+2}{2c_{|S|}} \left(1 - \left(\|x\|^S\right)^2\right), & \|x\|^S \leq 1 \\ 0, & \text{else.} \end{cases}$$

where $|S|$ denotes the dimensionality of the subspace and $c_{|S|} = \frac{\pi^{|S|/2}}{\Gamma(|S|/2+1)}$ is the volume of the $|S|$-dimensional unit sphere; gamma function $\Gamma(n+1) = n * \Gamma(n), \Gamma(1) = 1, \Gamma(1/2) = \sqrt{\pi}$ used in the volume computation. Each kernel is scaled in width according to a *bandwidth* ε which corresponds to the area of influence of a density based subspace clustering algorithm. For subspace clustering, we need only the Epanechnikov kernel weights $1 - \left(\|x\|^S\right)^2$ to obtain the following weighting function according to Definition 1 :

Definition 3. Epanechnikov Density Measure
Let $\mathcal{W}(t) = 1 - t^2$ be the Epanechnikov weighting function. We define the Epanechnikov density measure for an area of influence specified by ε as:

$$\varphi(o) = \sum_{p \in \mathcal{N}_\varepsilon^S(o)} \left(1 - \left(\frac{\|o-p\|^S}{\varepsilon}\right)^2\right)$$

As seen above, we can remove dimensionality bias by taking the expected density for subspaces into account. As clustering aims at detecting dense regions in a given data set, clusters should have higher density values than data without any clusters. A data set without clusters corresponds to uniformly distributed data, i.e. all values are taken with the same probability. By requiring that density should exceed the expected

density of uniformly distributed subspaces, we ensure that no pseudo-clusters are "detected". By applying this on the Epanechnikov density measure φ we obtain the unbiased Epanechnikov density measure:

Definition 4. Unbiased Epanechnikov Density Measure
The unbiased density measure for the Epanechnikov influence function φ is given by

$$\frac{1}{\alpha(S)}\varphi(o) \quad \textit{with}$$
$$\alpha(S) = E_S\left[\varphi(o)\right] = \frac{2n\varepsilon^{|S|}c_{|S|}}{v^{|S|}(|S|+2)} \tag{2.1}$$

with $c_{|S|}$ the volume of an $|S|$-dimensional unit sphere and $v^{|S|}$ the overall volume of an $|S|$-dimensional subspace in a universal domain $[0, v]$.

Dimensionality bias can be removed for other kernel density estimators as well by normalizing the density measure with the reciprocal expected density. The statistical approach has two advantages: the effectiveness of kernel estimators has been studied in theoretical and practical settings, and computation of the expected density for density functions follows standard methods.

2.3 DUSC subspace cluster definition

Intuitive density threshold. The density threshold is a core parameter since it sets the dividing line between dense objects and noise. As this parameter has to be set by the user it is important for users to have an intuitive understanding of this parameter. Commonly, users do not know density distribution apriori, which makes the choice of a density value difficult. We exploit the fact that in our approach density is measured with respect to the expected density as discussed before. Consequently, users do not need to specify absolute density thresholds, but only a factor by which the expected density has to be exceeded. Following the definition in the previous section, an object o is dense in subspace S according to the expected density $\alpha(S)$ iff:

$$\frac{1}{\alpha(S)}\varphi(o) \geq F$$

where F denotes the density threshold. As the density factor F is independent of the dimensionality and data set size, it is much easier to specify than traditional density thresholds. Moreover, we demonstrate in the experiments that this parameter is robust with a setting of $F > 50$ for many applications.

Empty space problem. In high dimensional settings another problem occurs which has to be considered by density-based subspace clustering algorithms. With increasing dimensionality the expected density and hence the expected number of objects contained in an area of influence drops exponentially [BGRS99]. For high dimensional subspaces this would result in almost zero expected objects and a meaningless

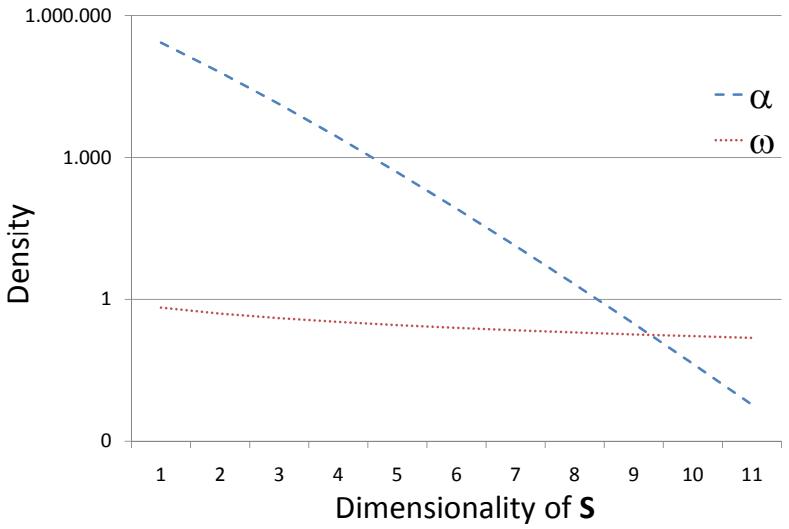


Figure 2.1: Density thresholds α and ω

density estimation. In statistics, this effect is termed "empty space problem" [Sil86]. Compared to the expected density an object may be determined as dense even if the area of influence is nearly devoid of observations, resulting in pseudo-dense single objects. In low and middle dimensional subspaces the expected density ensures unbiased density estimation. However, for high dimensional subspace, where the considered subspace itself falls prey to the curse of dimensionality ($E_S[\varphi(o)] \to 0$), every object tends to become denser than expected. To remove such pseudo-dense objects we introduce a specific density constraint on the expected density of η objects in the area of influence. This constraint ensures that the density value of a dense object is not only F times more dense than expected, but additionally exceeds the density obtained by η objects in its area of influence.

The expected density value of an object o which contains η objects in the area of influence $E_\eta\left[\frac{1}{\alpha(S)}\varphi(o)\right]$ can be derived as follows:

$$\begin{aligned} \frac{1}{\alpha(S)}\varphi(o) &\geq E_\eta\left[\frac{1}{\alpha(S)}\varphi(o)\right] \\ \varphi(o) &\geq \eta\frac{2}{|S|+2} \\ \varphi(o) &\geq \eta \cdot \omega(S) \qquad \left[\omega(S) := \frac{2}{|S|+2}\right] \end{aligned} \tag{2.2}$$

To guarantee that objects are not considered dense if the ε sphere is virtually empty, a very small value for η is sufficient (generally two or three). Intuitively one requires each object to contain at least η objects in its area of influence. Thus, even if the expected density drops to almost zero, this second constraint ensures a meaningful density notion. Users typically do not need to change this value.

Figure 2.1 illustrates the effect of increasing dimensionality on the two density thresholds α and ω [for parameter setting $\eta = 2; \varepsilon = 0.2; v = 1; n = 100000$]. The expected density α decreases exponentially with the dimensionality, while ω (the

expected density of η objects) only depends linearly on the dimensionality of the subspace (please note the logarithmic scale). The very low value for α in higher dimensions confirms the *empty space* problem. The α threshold thus applies to most dimensionalities, while ω applies only for high dimensional subspaces.

Our new density-based subspace clustering model below combines the density constraints α and ω given in formulae (2.1) and (2.2). The combination of α and ω ensure an unbiased density notion without defining objects in nearly empty regions as dense.

Cluster Size. So far, we have studied the density of individual objects. Subspace clusters, following density-based clustering paradigm, are connected sets of such dense objects. To ensure that clusters reflect the inherent structure of the data, they should contain a certain minimum number of objects. This constraint $minSize$ is typically about 1% of the database size. Thus, we can ensure to find clusters with sufficient many objects representing a quite large part of the data. We actively exclude clusters with a size of less than 1% of the data as these small pseudo clusters would not provide a sufficient contribution to the overall clustering result.

Redundancy. While the cluster size is a first criterion to select the final clustering result, redundancy is a more complex criterion to be checked for subspace clustering. Since the number of possible subspace projections is exponential in the number of dimensions, subspace clustering algorithms often produce numerous redundant subspace clusters. To avoid excessive cluster outputs which contain essentially the same information repeated in different dimensionalities, we check if a cluster O in subspace S is redundant. As a first redundancy notion for subspace clustering, DUSC provides a basic definition to exclude redundant clusters. We define a cluster as redundant if (most of) the objects contained in the cluster are also contained in another cluster in a higher dimensional subspace $S' \supset S$. Intuitively we keep only the highest dimensional cluster and remove all of its projections if they show up as redundant views on the same set of objects. We use a parameter R to specify the degree of redundancy acceptable to the user. A cluster in a lower dimensional projection is only acceptable as a non-redundant cluster, if only a fraction R of its objects are already covered by a higher dimensional cluster. Please note, that more details about our redundancy definition are given in the following chapters. As the first non-redundant subspace clustering model, DUSC provides a quite simple redundancy definition. Further extensions will be a major topic in this thesis, as well as the use of redundancy elimination for efficiency enhancement in subspace clustering.

The resulting subspace cluster model taking these conclusions into account is formalized in the following.

Definition 5. DUSC Subspace Cluster $C = (O, S)$
A set of objects $O \subseteq DB$ in subspace $S \subseteq DIM$ is a subspace cluster if:

- *objects in O are S-***connected***:*
  $\forall o, p \in O : \ \exists k : \ \forall i = 1, \ldots, k-1 : \ \exists \ q_i \in O :$

$\|q_i - q_{i+1}\|^S \leq \varepsilon \wedge q_1 = o, q_k = p$

- **more dense than expected and not pseudo-dense:**
  $\forall o \in O$: $\varphi(o) \geq max\{F \cdot \alpha(S), \eta \cdot \omega(S)\}$
- O *is* **maximal**, *i.e. contains all S-connected objects*
  $\forall o, p \in DB$, o, p *S-connected* $\Rightarrow (o \in O \Leftrightarrow p \in O)$
- **minimum cluster size:** $|O| \geq minSize$
- **not redundant:** $\neg\exists(O', S')$ *density-based subspace cluster with*
  $O' \subseteq O \wedge S \subset S' \wedge |O'| \geq R \cdot |O|$

The DUSC subspace clustering model extends existing density-based notions of maximality and connectedness with statistically sound density computation via normalized Epanechnikov kernel and expected density. Furthermore, each cluster contains a large part of the data and is not redundant. Overall, DUSC proposes the first unbiased and non-redundant subspace clustering model. In addition to this high quality model we will provide general processing schemes for efficiency enhancement in Part II. In the following we will only briefly show specific monotonicity properties for the novel DUSC model, while a more generalized perspective on efficient computation is provided in the second part of this thesis.

2.4 Derived monotonicity for DUSC model

As subspace clustering mines clusters in high dimensional data spaces, computing the clusters based on the DUSC model in a naive way is infeasible as the number of possible subspaces (and subspace clusters) is exponential with the dimensionality. Thus, we propose a monotonicity property for efficient and lossless pruning based on the DUSC model. Based on this pruning criterion DUSC is practically applicable just like previous methods [AGGR98, KKK04].

In general, pruning of subspace regions requires a monotonicity on some property of subspace clusters. A region which does not form a subspace cluster in some dimensionality implies that this region cannot be a subspace cluster in any higher dimensional subspace, and we may safely prune this region from further consideration. As the density definition given in Definition 5 depends on the dimensionality of the analyzed subspace, it is not monotonous in the above sense and thus cannot be used directly for pruning. The higher dimensional a subspace, the lower its expected density. Thus, a region which is not dense according to a low dimensional subspace's density threshold, may be dense with respect to a higher dimensional subspace's threshold. To overcome this problem, we introduce the new concept of weak density threshold.

Definition 6. Weak density

An object o in a subspace S is defined as **weak dense** *if:*

$$\varphi(o) \geq max\{F \cdot \alpha(DIM), \eta \cdot \omega(DIM)\}$$

The weak density definition uses the highest, $|DIM|$-dimensional threshold $max\{F \cdot \alpha(DIM), \eta \cdot \omega(DIM)\}$ for the density of an object o in subspace S. Thus if an object o is not weak dense, o cannot be dense in any super subspace of S. Moreover, density-connected sets can be pruned if they contain less than $minSize$ objects. Pruning based on these two properties, called *weak monotonicity*, is used by the DUSC algorithm to efficiently prune the search space. It is valid in the sense that no cluster is wrongfully dropped from consideration.

Based on our weak density criterion we can use any bottom-up processing (used e.g. in CLIQUE or SUBCLU [AGGR98, KKK04]) for efficient computation of our DUSC model. The monotonicity of our weak density ensures effective pruning. As soon as one reaches a region that does not fulfill the weak density criterion, all higher dimensional projections of this region can be pruned. Thus, our enhanced density-based subspace cluster definition might be plugged into existing subspace clustering algorithms.

However, as such algorithms are based only on simple pruning heuristics, we have developed further efficiency improvements for computation of the DUSC model. We will discuss these in Part II where also other general processing schemes for efficient subspace clustering are proposed. The first is generally applicable for density-based subspace clustering and will be described in Chapter 5. The second is based on a novel redundancy pruning and proposes a novel index structure described in Chapter 6.

2.5 Experiments

We ran extensive experiments on real world data (Pendigits, Glass, Vowel [AN07] and Shapes [KWX+06]) to demonstrate the quality of the DUSC subspace clustering model. While in this chapter we provide only preliminary experiments to highlight the quality of the DUSC model, further experiments are provided for efficient algorithms based on the DUSC model in Chapter 5 and Chapter 6.

Quality of the subspace clusters is determined in terms of purity and coverage.

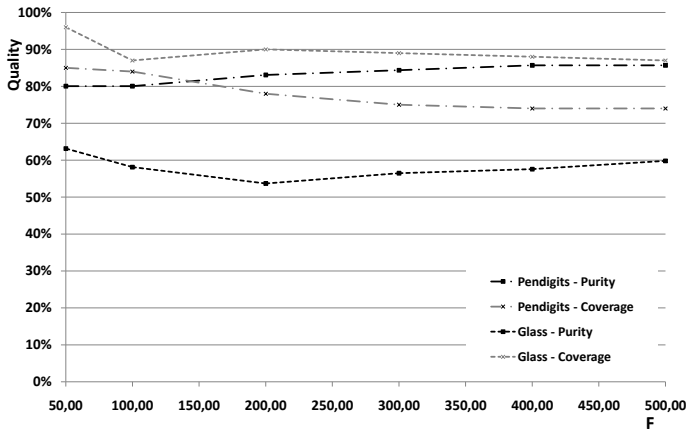


Figure 2.2: Quality vs. density threshold F

	DUSC 0%		DUSC 5%		DUSC 10%		SUBCLU		SCHISM	
	P	C	P	C	P	C	P	C	P	C
Pendigits	86	74	83	87	81	92	58	100	77	100
Glass	60	87	51	90	50	93	44	100	44	99
Vowel	82	70	79	100	74	100	10	100	42	100
Shape	100	31	100	31	100	31	98	82	100	1

Table 2.1: Cluster quality for real data sets

While coverage measures the size of the clustering, purity is determined using entropy, i.e. $H(O) = -\sum_{i=1}^{k} p(i|O) \cdot \log(p(i|O))$ for k class labels in cluster O. For a set of clusters we take the average entropy weighted by the number of objects per cluster. For readability, we take the inverse entropy and normalize it to a range of 0% to 100% by dividing by the maximum entropy ($1 - H(O)/log(k)$). Coverage is the percentage of objects in any subspace cluster. It indicates the ratio of clustered objects to noise. The amount of noise in a data set is typically not known apriori, but noise is present in most real world data sets. As sparsely populated regions often exhibit a weak correlation to the class label, cluster purity can be improved if less objects belong to a cluster (coverage drops). Thus we evaluate clustering quality by purity and coverage in combination. Our algorithm has few and intuitive parameters. They can be easily understood by users and are very robust on different data sets. Recall that $minSize$ of a cluster is the minimum number of objects per subspace cluster. Values around 1% of the data lead to manageable result sizes. Lower values produce more subspace clusters than users might want to study, whereas higher values diminishes the output size. η is fixed to two to eliminate the empty-space problem. Obviously, at least two objects should be contained even in very high dimensional subspace clusters. The density-thresholds α and ω are directly computed from the expected density. Users only provide a factor F which describes by how much the expected density should be exceeded. This F is independent of the database size and its dimensionality. Finally, the bandwidth ε regulates the kernel density. This parameter can be estimated using standard methods from statistics [Sil86].

We demonstrate robustness of our DUSC algorithm on two real world data sets with very different data distributions: Pendigits and Glass. The first experiment studies the effect of density threshold F on purity and coverage. The results shown in Figure 2.2 confirm that DUSC is remarkably robust with respect to F. Furthermore, we evaluated the overall quality [purity (P) and coverage (C)] of DUSC against SUBCLU and SCHISM using real world data sets. For DUSC we used the default values for F according to the heuristic in Section 2.3. As SUBCLU and DUSC are both density-based clustering algorithms we used the heuristic presented in [KKKW03] to determine ε. SCHISM is a grid based approach. Its parameters ξ, τ are also determined using the original heuristic in [SZ04]. For their third parameter u which cuts off low dimensional clusters, we noticed that the heuristic does not yield good results in terms of cluster quality. To obtain better quality results for SCHISM that explain more about the true differences between different density models, we used an even lower value for u than suggested by the authors. However as SCHISM is a grid based

approach it still does not reach the quality of DUSC. The first column in Table 2.1 shows the quality results of DUSC with redundancy set to zero which are the best measured qualities for all data sets. Allowing more redundancy, coverage increases significantly and purity goes only slightly down. However, even for $R = 10\%$ DUSC shows better purity than the competing algorithms. The fact that coverage is not 100% indicates that DUSC can distinguish between noise and clusters in subspaces of varying dimensionalities. The pendigits data set, for example, contains handwritten numbers, some of which are clearly different from the rest of the data set. Biased algorithms like SCHISM and SUBCLU do not detect noise, but assign all objects to clusters. The last data set SHAPE contains rotated versions of 9 different shapes, but only 3 of the shapes clearly form clusters. Thus most of the objects have to be considered noise. DUSC detects the given clusters correctly while SCHISM detects only a small part of the clusters and SUBCLU mixes up clusters with noise (less than 100% purity).

2.6 Benefits out of an unbiased density-based model

We introduced DUSC, an adaptive density-based subspace cluster model. Using both statistical density estimation and expected density in varying subspaces, we are capable of accurately grasping the inherent data structure without dimensionality bias. As shown in our experiments DUSC outperforms other subspace cluster models in terms of cluster quality. Thus, we use this fundamental model in most of our approaches. In all of these methods we check the density of an object against the expected density for the considered subspace (cf. Challenge 1). For our adaptive subspace cluster models we first keep the neighborhood for density estimation fixed to ε and vary the density threshold. However, please note that in our more enhanced adaptive density measure described in Chapter 3 we include an adaptive neighborhood as well. We consider variable neighborhoods $\varepsilon(S)$ depending on the dimensionality of the considered subspace.

While DUSC is the first model proposed for non-redundant subspace clustering, we extended this fundamental idea to more enhanced models. In the following chapters we develop optimization techniques for the final subspace clustering result and show that further redundant clusters can be eliminated to enhance the overall clustering quality. Thus, our optimization models described in Chapter 3 and Chapter 4 tackle the last open parts of Challenge 3. Furthermore, in the second part of this thesis we propose efficient processing schemes based on our DUSC model.

Chapter 3

Relevant subspace clustering results

Subspace clustering aims at detecting clusters in any subspace projection of a high dimensional space. As the number of possible subspace projections is exponential in the number of dimensions, the result is often tremendously large. Traditional approaches fail to reduce results to relevant subspace clusters. Their results are typically highly redundant, i.e. for a single hidden cluster traditional approaches might output up to $2^{|DIM|}$ many redundant views on the same set of objects. While DUSC proposes the first method for non-redundant subspace clustering as described in the previous chapter, the underlying redundancy definition has its drawbacks as it considers only a pairwise comparison of clusters. Such local redundancy elimination is unable to remove redundancy induced by multiple subspace clusters. Thus, in worst case scenarios where redundancy is not induces by a single higher dimensional cluster but by multiple clusters, this local redundancy model fails to reduce the exponential result size.

In this work, we propose a novel model for relevant subspace clustering (*RESCU*), detecting the most interesting non-redundant subspace clusters. In contrast to previous local redundancy notions, our global redundancy elimination checks the relevance of each cluster against all other possible subspace clusters. It aims at reporting as few clusters as possible to reduce redundancy, yet cover as many interesting concepts as possible. Especially, it allows overlapping subspace clusters, i.e. objects being part of multiple clusters. Thus, the general aim is to cover almost all objects (except of noise) by interesting clusters. Each cluster has to contribute at least some additional objects, which is technically realized by an optimization of the result set based on both the users specified interest and our global redundancy notion. We prove that computation of this model is NP-hard. For *RESCU*, we propose a relaxation of our model that shows high accuracy with respect to our relevance model. Thorough experiments on synthetic and real world data show that *RESCU* successfully reduces the result to manageable sizes. It reliably achieves top clustering quality while competing approaches show greatly varying performance. Compared to our previous work on non-redundant subspace clustering we could further reduce the result size and increase the overall quality of subspace clustering.

3.1 Motivation and comparison with related work

Subspace clustering detects clusters in arbitrary projections by automatically determining a set of relevant dimensions for each cluster [PHL04, KKZ09]. Thus, one is able to detect objects as part of various clusters in different subspaces. This general requirement of objects grouping in multiple different subspaces can be motivated out of various application scenarios. In bioinformatics, for example, genome data analysis clusters genes (objects) which show similar expression levels in a subset of experimental medical treatments (attributes). Such similarities might indicate multiple functional relationships. Thus, each gene might appear in multiple roles (subspace clusters). In general, subspace clusters might overlap in the sense that they share objects. Recent research has seen a number of approaches using different definitions of what constitutes a subspace cluster [AGGR98, KKK04, SZ04, KKRW05]. As summarized in our evaluation study in Chapter 10, their common problem is that the output generated is typically huge.

As in general, subspace clustering allows multiple clusters per object, it has to cope with detection of exponentially many subspace clusters in arbitrary projections. Many of these clusters do not provide any further information, as more or less the same object groups are detected in multiple projections of the data. Such redundant subspace clusters should be removed and only the most interesting ones, which provide novel knowledge about the data should be reported.

In contrast to subspace clustering, projected clustering assigns each object to a single projection [AWY+99, MSE06]. This strict partitioning of the data into projected clusters can be regarded as extreme redundancy elimination. Projected clustering results in a manageable number of clusters, but is not able to detect overlapping clusters. Based on disjoint partitions of the data, projected clustering is only able to detect each object in at most one cluster. Subspace clusters that overlap in their objects are not detectable. Especially in applications where each object is represented by multiple clusters, projected clustering fails to detect all hidden clusters. It only provides one view on the data while the majority of hidden clusters in different views is not detected. Thus, projected clustering can be seen as a very restrictive cluster definition, by far to restrictive for the desired detection of overlapping subspace clusters in many of todays applications.

In our previous work we have proposed a novel subspace clustering definition removing redundant subspace clusters. However, we defined non-redundant and possibly overlapping subspace clusters only with a local scope. In our previous model a subspace cluster is redundant if it shares a certain fraction of objects with another one. Retaining only maximal subspace clusters, i.e. the highest dimensional one of two, results in a clear increase in clustering quality. This is due to the fact that maximal clusters tend to contain less noise and thus represent the inherent data structure more faithfully. This definition of redundancy, however, is limited in two respects. First, redundancy is based only on a pairwise comparison of clusters. And second, the redundancy check incorporates only the fraction of jointly detected objects. We call this a local redundancy definition, as only local properties like object count and comparison of two subspace clusters is used. Consequently, a

redundant subspace cluster that is covered by a combination of high dimensional subspace clusters is still reported as non-redundant. In contrast, we aim at a global redundancy check including a more flexible interestingness definition comparing each cluster with the overall set of detected subspace clusters.

A recent approach aims at extracting non-redundant axis-parallel subspace regions [MS08]. It defines non-redundant results as the minimal subset of subspace clusters to approximately compute the support of any other subspace cluster (assuming uniform distribution otherwise). This statistical approach, however, is based on the assumption of uniform distribution inside a cluster. Moreover the redundancy model is limited to the fixed cluster definition. As we show in our evaluations all of these approaches fail to detect all and only the hidden concepts, i.e. clusters, in a high dimensional database.

Our goal is to derive a novel model for non-redundant subspace clusters that takes a global look at overlapping clusters. The aim is to find all and only relevant concepts by optimizing the overall clustering result. The main contributions of our work are:

- Detection of *all* interesting clusters, but without the overwhelming result size of subspace clustering.
- Detection of *only* non-redundant clusters, but without the strict partitioning of projected clustering.

To achieve these objectives, we introduce a new global relevance model for subspace clustering. We combine a user specified interestingness function for subspace clusters with our novel coverage criterion for an overall redundancy removal. By including the most interesting clusters and excluding redundant clusters, our result set contains all and only relevant subspace clusters. Any object can be part of multiple clusters, allowing overlapping subspace clusters. It is desirable to report as few clusters as possible to reduce redundancy, yet cover as many interesting concepts as possible. Furthermore, our relevance model is neither restricted to one cluster definition nor based on a fixed interestingness rating. These two aspects are typically application dependent and can be adapted by the instantiation of our model. Considering computation complexity, we prove that our relevant subspace clustering is NP-hard. Thus, for an efficient computation of our relevance model we propose an approximative algorithm generating possible cluster candidates in best-first order according to their relevance.

3.2 Relevant subspace clustering

In this section, we introduce our relevant subspace clustering definition. Most existing approaches use only local properties, i.e. objects O and dimensions S, to define if $C = (O, S)$ is a subspace cluster or not. We take a global view, considering the theoretic set of possible subspace clusters $ALL = \{C_1 \dots C_n\}$ in total for our relevant clustering definition. As pre-computation of ALL is too expensive we propose an

approximative algorithm using a novel on-demand cluster generation derived from our relevance model in Section 3.3. Most of the clusters in ALL do not provide any knowledge for the user, thus, our relevance model defines which of these clusters to output (cf. Fig. 3.1). We aim at reducing the output to only the *relevant* subspace clustering $M \subseteq ALL$. The remaining clusters overwhelm the user, thus hinder the analysis and are removed.

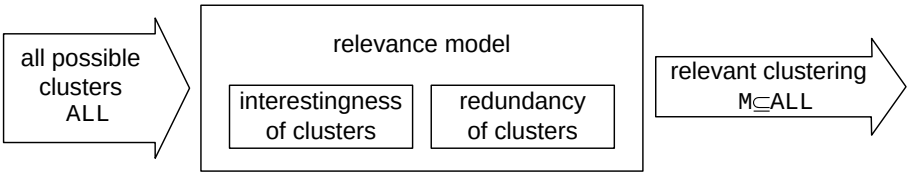


Figure 3.1: Components of a relevance model

Two aspects are important for relevance. One is the user specified *interestingness* of a cluster itself. The interestingness evaluates a cluster locally via its properties like dimensionality or size. For example, clusters in only one dimension might not be interesting for the user. Interestingness is often user or application dependent and therefore it should be exchangeable. A detailed discussion of our flexible interestingness model is given in Section 3.2.1.

Another aspect is the *redundancy* of clusters. Redundancy means that this cluster does not contribute to new knowledge with respect to other clusters (for example if another cluster with similar properties exists). Consequently, a redundant cluster should not be output. The redundancy is discussed in Section 3.2.2.

Please note that the two aspects *interestingness* and *redundancy* are not entirely independent. Both aspects are interleaved in our relevance model: A relevant cluster is interesting but not redundant. Thus, our relevance model uses both, the user specified interest and our redundancy definition in an interleaved way to optimize the overall result. It includes the most interesting clusters but excludes redundant clusters of low interest. Thus, the user specified interest is used also in the redundancy elimination. Given the choice between two clusters with similar properties, the less interesting one should be marked as redundant. Our optimization excludes both clusters with low interest and clusters that cover to few novel objects (not yet covered by other clusters). This interleaved relevance model of user specified interest and coverage of novel objects is the key property of our relevance optimization. The overall relevance model is described in Section 3.2.3. Interleaved, it ensures both the maximal coverage of objects by interesting clusters and a sufficient coverage on novel objects in each cluster such that the result size is minimized.

In Section 3.2.4, we prove important complexity results and conclude in Section 3.2.5 with an instantiation of our model.

3.2.1 Interestingness of a cluster

In this section, we describe the flexible interestingness model of RESCU to quantify the interestingness of a single cluster. Figure 3.2 gives an example of clusters in

dimensions 1 and 2 (left), and 3 and 4 (right), respectively. Objects in both illustrations are represented using the same symbol. For example, assume that we deem clusters with more dimensions more interesting. The 2d (two dimensional) clusters C_1 and C_2 are then favored over the 1d clusters $C_3 - C_6$. The number of objects, the diameter or the density of a cluster are other typical choices of interestingness. All of these measures can be used by the user to provide a specific interest characterization. In contrast to this user driven specification, further measures using statistical properties of the data might also be applicable. Derived out of the data distribution such measures provide more objective interest specifications as widely used in other research areas such as frequent itemset mining [GH06, ST96]. As we provide a flexible interestingness model, these measures could also be used as instantiations of interest. However, as we focus more on the general perspective and the interleaved usage of interest in our relevance model, we instantiated interest with a quite simple user-dependent interestingness measure based on the size and dimensionality of a cluster.

For the general perspective, let us discuss how interestingness is currently used (or not used) in subspace and projected clustering. Interestingness is not handled explicitly in existing projected and subspace clustering algorithms. In projected clustering algorithms, overlapping clusters are not detected due to the partitioning. In Figure 3.2, the most interesting clusters C_1 and C_2 are not found. They both contain some objects that occur in the other cluster as well, and are therefore mutually exclusive, even though the objects occur in dimensions 1 and 2, or 3 and 4, respectively. As a consequence, potentially interesting clusters are missed by the occurrence of *other* interesting clusters. Another problem is the handling of outliers only in a post-processing step. Interestingness of the clusters is calculated before removal of outliers and hence misleading values can occur.

Subspace clustering algorithms realize a very limited interestingness calculation. Each set of points and dimensions that fulfills the cluster definition (e.g. exceeding a density threshold) is equally interesting. Thus, there is no specification of interest in subspace clustering. It is a binary decision "cluster" or "no cluster". The user is not able to controll what kind of clusters he is more interested in. After detection of all subspace clusters there is no additional selection of most interesting clusters out of the huge result set.

And finally, the calculation of the interestingness in both models is usually fixed and cannot be modified by the user. It is fixed to the formal definition given by the researchers that developed these clustering methods. Flexibility to include the user interest is limited by the different parameters provided by each model individually. A general and flexible interest specification providing the ability to instantiate the model by the application dependent interest is not provided by these model.

Our model overcomes these problems. We assign an interestingness value to each cluster. It is a local rating, so that other clusters do not influence this measure. Furthermore, our cluster definition (cf. Sec. 3.2.5) accounts for outliers so that misleading values do not occur.

As a general and flexible framework, we model the (un-)interestingness via a cost

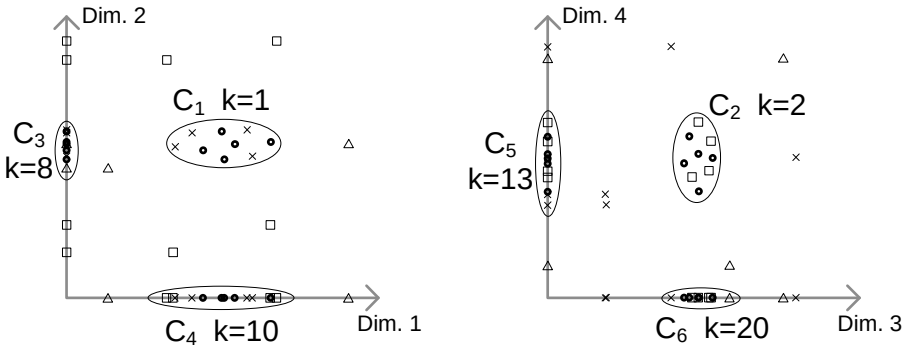


Figure 3.2: Interestingness by cost values

function k. Small values denote interesting clusters. For example, in Figure 3.2 the 1d cluster C_4 might get a cost value of $k = 10$ while the more interesting 2d cluster C_1 might get a lower cost value of $k = 1$.

For any cluster C, given by the set of its objects O and the corresponding dimensions S, we define the cost function as follows:

Definition 7. Cost function
Let $\mathcal{P}(DB)$ be the power set of all database objects and $\mathcal{P}(Dim)$ the power set of the dimensions. A cost function

$$k : \mathcal{P}(DB) \times \mathcal{P}(Dim) \to \mathbb{R}$$

assigns cost $k(O, S)$ to the subspace cluster $C = (O, S)$.

By defining a cost function, we can account for several aspects of a subspace cluster, like the dimensionality or the density. Changes to the cost function yield an easy adaptation of the model. This flexibility is not achieved by other approaches. An instantiation of the function is presented in Section 3.2.5. By first calculating the interesting values individually and allowing overlap we find higher quality clusterings. In Figure 3.2 we may select both C_1 and C_2.

For computational reasons, we assume cost functions which assign a strictly positive value to all subspace clusters, i.e. given a set $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$ of clusters, the function fulfills $k(O_i, S_i) > 0$ for all $(O_i, S_i) \in M$. To mine the most interesting clusters we minimize the overall cost. However, to also maximize the number of objects covered by a clustering, we additionally take coverage into account. Together low cost and high coverage of novel objects are the main properties of a relevant subspace cluster. We formalize coverage as follows:

Definition 8. Coverage of a clustering
Given a clustering $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$, the coverage of M is defined as follows: $Cov(M) = \bigcup_{i=1}^{n} O_i$

The coverage of a clustering M is the union of the objects in all selected clusters. We now define the overall relative cost of a clustering as the sum of the individual cost values normalized by the number of covered objects.

Definition 9. Overall relative cost of a clustering
Let $M = \{(O_1, S_1), \ldots, (O_n, S_n)\}$ be a clustering and k a cost function. We define the overall relative cost of M as:

$$RK(M) = \frac{K(M)}{|Cov(M)|} \text{ with } K(M) = \sum_{(O_i,S_i)\in M} k(O_i, S_i)$$

The smaller the overall relative cost $RK(M)$, the more interesting is the clustering M per covered object. We achieve a high coverage and at the same time small total cost.

3.2.2 Redundancy of a cluster

The second aspect of relevance is non-redundancy. A large cluster C and all of its lower dimensional projections could be assigned low cost values if interestingness is based on size. Selecting all projections along with C based on interestingness alone leads to a poor overall result. One gets very many redundant clusters, while C would be sufficient.

We therefore take a global view for redundancy elimination and compare a cluster with other clusters. While the interestingness is a local measure based on the cluster itself, the redundancy takes other clusters into account.

Existing projected and subspace clustering algorithms do not address redundancy handling adequately. Projected clustering simply forces results to be non-redundant by assigning each object to a single cluster at the cost of missing overlapping clusters. Subspace clustering algorithms, in contrast, either use no or a mere local approach to check the redundancy. As proposed in Chapter 2 such an approach compares only two clusters. If the clusters cover nearly the same objects, one of them is redundant. The problem of this local approach is illustrated in Figure 3.3. Obviously, in both subfigures the cluster C_2 is redundant because it is induced by the other clusters C_1, resp. C_{1a}, C_{1b}. A local approach could identify the redundancy in the left figure. Cluster C_2 is redundant, as it covers C_1 and only a few additional objects. In the right figure, the fraction of points shared by C_{1a} and C_2 as well as by C_{1b} and C_2 is small, and the cluster C_2 is misleadingly classified as non-redundant. This mistake is the result of the local view on redundancy, i.e. for each check only a pairwise comparison of clusters is performed (cf. Chapter 2).

We use a global view for the redundancy checks, i.e. we use all clusters at the same time to judge the redundancy of another cluster. This approach results in more accurate decisions.

As one can see from the above example, the redundancy of a cluster is linked to the coverage of objects. If a set of clusters shares many objects with a cluster C, C is a redundant cluster. In other words: A cluster is redundant if it does not cover many new objects. The cluster C_2 in Figure 3.3(b) is redundant, because with respect to the two other clusters only a few new objects are covered. The same holds for the cluster C_2 in Figure 3.3(a). The fact that we consider all clusters for the redundancy checks yields a global redundancy model. Thereby we identify in both subfigures the cluster C_2 as redundant.

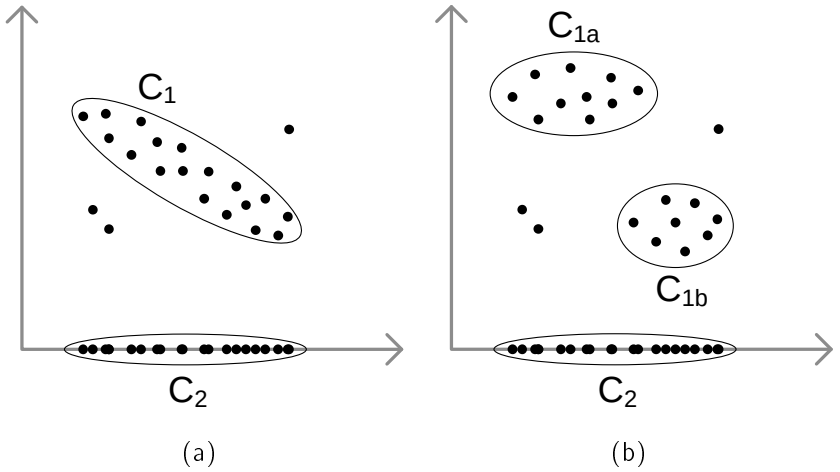


Figure 3.3: Local and global redundant clusters

Basic Set-Cover approach If we select the *minimal* number of clusters such that all objects are covered, we can realize a global redundancy model. The output of clusters that only contain already covered objects is prevented. This check is with respect to all other clusters in the result set. At the same time we identify multiple overlapping concepts in the data, because all objects have to be covered.

This novel way of subspace clustering can be considered as an instance of the Set-Cover problem [GJ79]. Given several finite sets, Set-Cover seeks for the minimal number of sets that cover the whole population. In the clustering context, we want to find the minimal number of clusters, such that all objects are covered.

However the direct application of Set-Cover to our task is not possible.

Problem 1: Simply choosing the minimal number of clusters means that usually low dimensional clusters, which tend to contain more objects, are preferred over high dimensional clusters. This preference usually conflicts with the user's notion of interestingness. Instead of choosing the minimal number of clusters, we determine the clustering with minimal relative cost (cf. Def. 9). This setup takes the desired interestingness notion into account. In Figure 3.3(b), for example, we would choose the two 2d clusters instead of the one 1d cluster. We thus use an extension of the Set-Cover problem to the Weighted-Set-Cover problem [Chv79].

Problem 2: Covering all objects by clusters is not always a meaningful solution, as some databases contain outliers that do not fit to any concept. The Set-Cover problem enforces a complete cover of all objects and potentially finds no solution in this case. Or, all objects can be covered but only by uninteresting clusters. For example, if the outliers are only contained in the 1d clusters the Set-Cover problem enforces choosing these uninteresting ones. An example is in Figure 3.3(b), as one has to select C_2 to get a complete coverage. We therefore propose a generalization of the Set-Cover problem to cope with outliers.

Gain-based redundancy We solve these problems by a gain-based extension of the Set-Cover problem. The basic idea is to measure the gain of a new cluster if we

add it to a known clustering. In other words, we have to answer the question: Is it worthwhile to take the "more complex" clustering? Our gain definition answers this question by measuring both redundancy w.r.t. the overall result set and including a weighting according to a user specified interestingness. Intuitively, our *cluster gain* measures the amount of novel knowledge (novel objects) covered by each cluster w.r.t. its interestingness given by the cost function. This is the gain we would obtain if we include this cluster into the result set.

Let us consider Figure 3.4 and assume the clusters C_1 and C_2 to be selected. Intuitively, the cluster C_3 is redundant with respect to the selected ones, so its gain should be small. Two important aspects contribute to this fact. First, the cluster C_3 covers only a few new objects, i.e. many objects are already contained in other clusters. These new objects, covered by the cluster $C_3 = (O, S)$, can be calculated via the residual set $O \backslash Cov(\{C_1, C_2\})$. And second, the cost of the cluster is very high, i.e. the cluster is not interesting. High cost $k(O, S)$ should correspond to a low gain. Overall we take the ratio of these two measures. Formally, *cluster gain* is the additional coverage that the cluster C provides to the overall result set M, in relation to its cost.

Definition 10. Cluster gain
Given a cluster $C = (O, S)$, a clustering M and a cost function k for $M \cup \{C\}$. The cluster gain of C with respect to M is:

$$clus\_gain(C, M) = \frac{|O \backslash Cov(M)|}{k(O, S)}$$

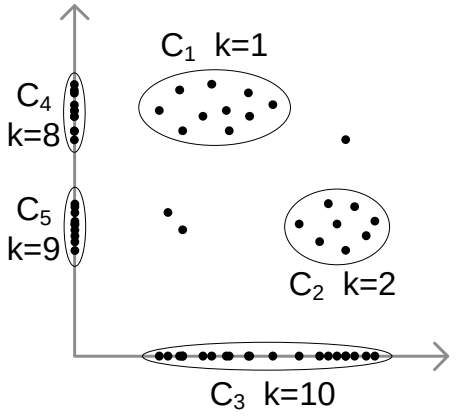
Usually, high dimensional clusters are favored so that these clusters should get low cost values. Nevertheless, by changing the cost function we can also change the cluster gains and hence the preferred clusters.

We identify a cluster as redundant (with respect to a given clustering) if its cluster gain is too small. In Figure 3.4 the cluster gain of C_3 (with respect to $\{C_1, C_2\}$) is only 0.3 as C_3 covers 3 new objects and $k(C_3) = 10$. Instead, C_2 has a higher cluster gain of 4 (with respect to $\{C_1\}$).
Consistent with this idea, a clustering M is redundancy-free if *all* clusters from M exceed a minimum cluster gain. The gain has to be measured with respect to the remaining clusters from M so that each cluster adds sufficient information to the overall clustering.

Definition 11. Redundancy-free clustering
Given a clustering $M \subseteq ALL$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. The clustering M is redundancy-free, iff

$$\forall C \in M : clus\_gain(C, M \backslash \{C\}) > \Delta$$

In this way we achieve a global view for the redundancy checks. Each cluster C has to compete with all remaining cluster $M \backslash \{C\}$ in the result set at the same time. As one can see, the clustering $\{C_1, C_2\}$ in Figure 3.4 is redundancy-free while the clusterings $\{C_1, C_2, C_3\}$ or $\{C_1, C_2, C_4\}$ contain redundant information (assuming $\Delta =$



$$clus\_gain(C_3, \{C_1, C_2\}) = \frac{3}{10} = 0.3$$
$$clus\_gain(C_4, \{C_1, C_2\}) = \frac{1}{8} = 0.13$$
$$clus\_gain(C_2, \{C_1\}) = \frac{8}{2} = 4$$
$$clus\_gain(C_1, \{C_2\}) = \frac{10}{1} = 10$$
$$\frac{K(\{C_1,C_2\})}{|Cov(\{C_1,C_2\})|} = \frac{3}{18} = 0.17$$
$$\frac{K(\{C_4,C_5\})}{|Cov(\{C_4,C_5\})|} = \frac{17}{21} = 0.81$$

Figure 3.4: Relevant clustering

0.5). Definition 11 gives us the basis to avoid redundancy in our clustering. However, if we remove clusters from a redundancy-free clustering, this property still holds for the smaller clustering. The removal of information, i.e. clusters, cannot generate redundancy. This means that also the empty clustering is redundancy-free. That is obviously not desired by the user. To obtain a relevant clustering, it should be redundancy-free but at the same time cover as many objects as possible.

3.2.3 Overall relevance of a clustering

To enforce the selection of clusters we introduce the property of concept-covering. A clustering M does *not* satisfy this property as long as clusters exist, which have a high gain and are not in M. M is not the result because further interesting concepts can be covered. A clustering is concept-covering, and hence the coverage sufficient, if all remaining clusters have a small cluster gain. We formalize this by:

Definition 12. Concept-covering clustering
Given a clustering $M \subseteq ALL$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. The clustering M is concept-covering, iff

$$\forall C \in ALL \setminus M : clus\_gain(C, M) \leq \Delta$$

Intuitively, a clustering is concept-covering if adding *any* new cluster *always* results in redundancy. A concept-covering clustering in Figure 3.4 is $\{C_1, C_2\}$. Clustering $\{C_1\}$ is not concept-covering because we could add C_2 without introducing redundancy.

The property of concept-covering clusterings is a relaxation of complete coverage. It solves the problem of enforcing a complete coverage as in the Set-Cover problem (cf. Problem 2).

RESCU: Relevant subspace clustering Our RESCU model demands a relevant clustering to be redundancy-free and concept-covering. However, as the example in Figure 3.4 illustrates, several clusterings could fulfill both properties, e.g. $\{C_1, C_2\}$

or $\{C_4, C_5\}$. To select the most interesting clustering, we additionally compare their relative cost (cf. Def. 9). Our relative cost function (cost per covered object) ensures finding an optimal clustering with both interesting clusters and high coverage. The relative cost for the clustering $\{C_1, C_2\}$ in Figure 3.4 is just 0.17, but 0.81 for $\{C_4, C_5\}$. This is formalized by:

Definition 13. Relevant subspace clustering (RESCU)
Given a clustering $M \subseteq ALL$ and a minimal cluster gain $\Delta \in \mathbb{R}^{\geq 0}$. M is relevant, iff

- *M is redundancy-free and concept-covering (Def. 11 & 12) AND*
- *M has minimal relative cost, i.e. $RK(M) \leq RK(N)$ for all redundancy-free and concept-covering clusterings $N \subseteq ALL$*

The relevant clustering in our previous example is thus $\{C_1, C_2\}$. Also for the example in Figure 3.2 the relevant clustering is $\{C_1, C_2\}$ even though the two clusters share some objects. This illustrates that handling of overlapping clusters is possible in our model.

The flexibility of our model additionally provides the possibility of classifying other clusterings as relevant by changing the interestingness criterion. If we adapt the cost function so that C_1 and C_2 in Fig. 3.4 get higher cost values, the clustering $\{C_4, C_5\}$ could become relevant. Thus our model enables the user to control the output as desired.

3.2.4 Complexity results

Calculating a RESCU clustering is an NP-hard problem. We prove this by giving a polynomial reduction of Weighted-Set-Cover problem, which is an NP-complete problem, to our RESCU model, i.e. Weighted-Set-Cover $\leq_p RESCU$. The Weighted-Set-Cover problem seeks those non-empty sets that together have minimal weights and fully cover all objects. For this reduction we need to map the input of the Weighted-Set-Cover problem to an input of RESCU and show that the resulting relevant subspace clustering is a valid solution for the Weighted-Set-Cover problem.

Theorem 1. *Computing RESCU (Def. 13) is NP-hard.*

Proof.
We show that Weighted-Set-Cover $\leq_p RESCU$.
Input mapping: We map the input of the Weighted-Set-Cover (objects, sets, weight function) to an input for RESCU (database DB, possible clusters ALL, cost function k). We further map the assumption of the Set-Cover (complete coverage exists) to ($\Delta = 0$) in RESCU. As RESCU is a more general problem it finds a clustering even in databases containing noise, where complete coverage is meaningless. We now have to show that relevant subspace clustering (cf. Def. 13) corresponds to a solution of the Weighted-Set-Cover problem.

RESCU generates a valid solution for Set-Cover:

(1) Every chosen set contributes at least one object to the overall coverage:
A relevant clustering M is non-redundant (Def. 11):
$\Leftrightarrow \forall\, C \in M : \mathit{clus\_gain}(C, M \backslash \{C\}) = \frac{|O \backslash Cov(M \backslash \{C\})|}{k(O,S)} > 0$
Thus, all sets contribute at least one object
$\forall\, C \in M : O = Cov(\{C\}) \not\subseteq Cov(M \backslash \{C\})$.
(2) For a Weighted-Set-Cover all objects have to be covered by at least one set:
A relevant clustering M fulfills the concept-covering property (Def. 12):
$\Leftrightarrow \forall\, C \in ALL \backslash M : \mathit{clus\_gain}(C, M) = \frac{|O \backslash Cov(M)|}{k(O,S)} \leq 0$
$\Leftrightarrow \nexists\, C \in ALL \backslash M :\ |O \backslash Cov(M)| > 0$
Thus, all objects are covered: $Cov(M) = DB$.
(3) The sum of weights is minimal for the chosen sets:
A relevant clustering M has minimal relative costs (Def. 13):
For all non-redundant and concept-covering clusterings
$N \subseteq ALL$: $RK(M) \leq RK(N) \Leftrightarrow \frac{K(M)}{|Cov(M)|} \leq \frac{K(N)}{|Cov(N)|}$
From $Cov(M) = Cov(N) = DB$ we have: $K(M)$ is minimal.
(1) $\wedge$ (2) $\wedge$ (3) $\Rightarrow$ M is a valid Set-Cover solution.
RESCU is NP-hard. □

As we have proven, RESCU is a generalization of the Set-Cover problem and thus NP-hard. Furthermore, it has two advantages for detection of relevant clusterings. First, we can handle outliers so that we can find a solution even if a complete coverage is not possible. And second, RESCU incorporates clustering properties and thus mines the most relevant concepts instead of simply covering the data.

3.2.5 Flexible instantiation of the model

By the flexibility of our model we can handle any definition of subspace clusters (as the input of the model) and cost functions (to rate the clusters). For a practical evaluation we instantiate these two aspects.

Definition of subspace clusters. We use density-based clustering because it detects clusters of arbitrary shape and size even in noisy data [EKSX96]. The idea is to define clusters as dense areas separated by sparsely populated areas. In this way outliers are not part of the clusters within the same subspace and misleading ratings are prevented. An object is considered dense if its neighborhood, i.e. an ε-distance region around it, is sufficiently populated. We follow the definition from [KKK04] with the modification, that we adjust the ε-neighborhood according to the subspace dimensionality. Thereby we account for increasing distances between the objects in higher dimensional spaces.

In contrast to the adaptation of the density threshold as proposed in Chapter 2, we increase the ε range such that several neighboring objects can be expected inside such a ε-range even in high dimensional subspaces. A fixed ε, as traditionally used in density-based clustering and also in previous subspace clustering approaches, tends to become empty with increasing dimensionality due to the empty space problem

mentioned in Section 2.3. Thus, even an unbiased density measure does not provide a meaningful density if ε-neighborhoods become empty. Using our previous Definition 5 simply does not provide any clusters in such high dimensional subspaces. For the instantiation of our RESCU model we propose to increase ε to tackle this challenge. The value of ε in a subspace with dimensionality d is

$$\varepsilon = \left[\frac{4 \cdot n}{3 \cdot \Gamma(1.5)}\right]^{\frac{1}{5}} \cdot \varepsilon_1 \cdot \left[\frac{d+2}{4 \cdot n} \cdot \Gamma\left(\frac{d}{2}+1\right)\right]^{\frac{1}{5}}$$

where ε_1 denotes the ε-range in the 1d subspace, n the database size and Γ the gamma function. Similar to the adaptation of density threshold in our previous work, we adapt the ε-neighborhood by using the expected density in the considered subspace. We derive our variable ε from a statistically optimal choice of ε provided by kernel estimators [Sil86] in a fixed dimensionality d. As we cope with arbitrary dimensional subspaces we propose an initial choice of ε_1 for 1d subspaces and scale this value with increasing dimensionality. The parameter ε_1 can be used to adapt our model to application dependent properties while the scaling to different subspaces is done automatically. The scaling factor is statistically motivated by an ε-sphere that prevents a constant expected density. In contrast to our previous work this adaptive density yields meaningful density values even for high dimensional subspaces and thus provides high quality results even in higher dimensional databases.

Definition of cost functions. The cost function used in our experiments is $k(O,S) = \frac{1}{|S|^\beta}$ with $\beta \geq 0$. Higher dimensional clusters get lower cost values and are therefore more interesting than lower dimensional clusters. Such a preference for higher dimensional clusters has also been used in our previous models. Variation of β influences how much the different subspace dimensionalities affect the cost value. A very high β-value implies that the result set mainly contains high dimensional clusters, whereas a low value tends to lead to lower dimensional clusters. If all cost values are nearly identical low dimensional clusters are usually preferred because these clusters generally cover more objects and so their cluster gain is higher.
We implemented further cost functions that reflect other interestingness objectives. For example, the density of the subspace clusters can be taken into account. Depending on the application scenario such cost functions can be used to adapt our flexible model to different interestingness definitions for the desired subspace clusters.

3.3 Relaxation of our model for efficient computation

There are two major challenges for efficient computation of relevant subspace clustering. First, as we have shown in Section 3.2.4, global optimization of the clustering is an NP-hard problem. And second, the assumed input set ALL is too large and its enumeration may be difficult or even impossible. Thus, we assume that there exists no efficient and especially no scalable solution. However, we provide an approximate solution tackling both of these challenges based on three main contributions:

- Greedy selection of clusters.
- Relevance update for concept-covering.
- On-demand generation and ranking of subspace clusters according to their cluster gain.

A naive approach would compute all possible subspace clusters (ALL is exponential w.r.t. number of dimensions) and then choose an optimal subset of clusters (exponential in the number of results). We use an approach that generates promising clusters on-demand and ranks them according to their relevance information. Thus we avoid an expensive pre-computation of all possible subspace clusters. With our greedy approach we relax the optimization by choosing in each step the most promising cluster available. Please note that a new cluster changes the overall coverage of the data, and it changes the relevance of remaining clusters. Updating the relevance is thus essential for concept-covering.

Greedy Selection of Clusters Our greedy approach iteratively includes the best cluster so far. Such an approximation idea is known for other NP-hard problems like Set-Cover [GJ79, Chv79]. For our novel relevant subspace clustering, this basic idea of greedy processing leads to efficient computation but also high quality subspace clustering results by choosing in each step the most relevant cluster according to our cluster gain definition.

By iteratively picking clusters, we relax two parts of Definition 13. First, instead of checking the global redundancy (Def. 11) we compute an approximative solution, as in each step i the relevance of a new cluster C_i is checked only w.r.t. clusters $M_i = \{C_1 \dots C_{i-1}\}$ already chosen in the previous $i-1$ steps. The cluster C_i is non-redundant w.r.t. M_i. Secondly, we choose the cluster with the highest *clus_gain* in order to have the most interesting clusters in the result and to approximate the minimal relative cost of the relevant clustering:

Definition 14. Relaxation of relevant clustering
C_i is inserted into the current result set M_i iff

(1) *C_i is a non-redundant cluster w.r.t. M_i:*
$clus\_gain(C_i, M_i) > \Delta$

(2) *C_i is the most interesting cluster in step i:*
$\forall C \in ALL \setminus M_i : clus\_gain(C_i, M_i) \geq clus\_gain(C, M_i)$

Our relaxation yields an efficient processing. It computes a chain of most relevant clusters C_i yielding an approximative solution for our RESCU model. It is a best-first method as in each step the cluster with the highest gain is selected. The greedy processing terminates if no more relevant clusters are available in the residual set of clusters according to the concept-covering property (cf. Def. 12).

Relevance Update The set of resulting clusters directly influences the cluster gain of the next cluster to be chosen. We have to update the cluster gain (Def. 10) of all candidates $C \in ALL \setminus M_{i+1}$, each time we insert a cluster C_i into the result set $M_{i+1} = M_i \cup \{C_i\}$. As the new cluster C_i changes the overall coverage of objects $Cov(M_{i+1}) \supseteq Cov(M_i)$, we have to adjust the relevance of all remaining cluster candidates. Our relevance update decreases the gain of redundant clusters that are already covered by C_i. Consequently, other concepts not yet covered have a relatively higher likelihood of being chosen in the following iteration.

Example: In our example given in Figure 3.4 we first choose C_1 as the first relevant subspace cluster with maximal cluster gain $clus\_gain(C_1, \{\}) = 10$. Intuitively, our algorithm chooses according to the cluster gain definition, which prefers higher dimensional clusters that contain objects not yet covered by other relevant clusters. Thus, in the second step we choose C_2 with the currently highest gain $clus\_gain(C_2, \{C_1\}) = 4$, a 2d-cluster with not yet covered objects. Choosing $\{C_1, C_2\}$ forces a relevance update as now most of the objects in C_3, C_4, C_5 are already covered. Thus, the next most relevant cluster is C_3 with an updated relevance of only 0.3 as it contributes only 3 objects to the overall clustering. Thus, assuming $\Delta = 0.5$, the algorithm has detected all and only the relevant clusters as any remaining cluster has a lower cluster gain.

On-Demand Generation and Ranking For our greedy processing we maintain an up-to-date ranked list of subspace clusters with a high cluster gain. However, it would be computationally too expensive to compute all possible clusters ALL and then sort them according to their cluster gain. In contrast to such an exhaustive generation of all possible clusters, our approach computes candidates on-demand and reduces computation to only the most promising regions. As most of the exponentially many clusters in ALL are not interesting or they are redundant w.r.t. the final result set, they do not affect the global optimization. Hence, our on-demand candidate generation computes only the most promising cluster candidates according to their cluster gain.

For on-demand generation of these regions we use our recent density estimation technique which combines a given set of lower dimensional cluster candidates (initially 2d clusters) to possibly interesting higher dimensional patterns (cf. Chapter 8). These new cluster candidates are inserted into our ranking based on their cluster gain. Due to relevance updates in later steps, the positions of these new cluster candidates might be rearranged. The currently most relevant candidates are at the top of the ranking.

Furthermore, as density-based clustering is a computationally expensive task we compute density estimation based on discretized grid cells to approximate the true densities (cf. Chapter 5). Efficiency is ensured as we only perform our density-based clustering on the top ranked candidates. Thus, by choosing the top candidate from the ranking, we focus in each step on the most promising cluster either to generate new candidates or to refine an approximative cluster. For inclusion in the clustering result according to greedy processing we select the top ranked cluster that has been

refined and used for new candidate generation.

Overall, using this on-demand ranking in our greedy approach, we ensure efficiency by generation of only a small set of promising subspace cluster candidates as needed. Furthermore, in each step we select the most promising (top ranked) region for processing. For further details about algorithmic enhancements please refer to Part II of this thesis.

3.4 Experiments

We compare RESCU with recent representatives of different high dimensional clustering paradigms: Grid-based subspace clustering CLIQUE [AGGR98] and its extension SCHISM [SZ04]; Density-based algorithms SUBCLU [KKK04], INSCY (cf. Chapter 6, as an efficient algorithm for the DUSC model presented in Chapter 2) and the approximative FIRES [KKRW05]; Projected clustering PROCLUS [AWY+99] and statistical P3C [MSE06] and StatPC [MS08].

Note that we have optimized parameters for each algorithm on each data set. Furthermore, we provide supplementary material[1] (executables, exact parameter settings and data sets used in our evaluation) for repeatability and comparison. For a fair comparison of the competing approaches we use our evaluation framework as described in Chapter 10. All implementations are in Java and experiments were run on Intel Core 2 Duo computers with 3 GHz and 2 GB main memory.

Benchmark data from the UCI archive [AN07] (also used in the evaluation study in Chapter 10) is used to study performance on real data. In addition, we use 17-dim. features as extracted in our previous work from sequence data in [KWX+06]. Besides the UCI 16-dim. pendigits data, we use 32 and 48-dim. variants by interpolation of the available polylines. As the true number of clusters is unknown for real data, we measure accuracy of class labels as in [MSE06]. We employ synthetic data for validating that all generated clusters are identified and for scalability experiments. Following the method in [KKK04] for overlapping density-based clusters in arbitrary subspaces, we generate data of different dimensionalities and hide subspace clusters with a dimensionality of 50%, 60% and 80% of the data dimensionality.

We measure quality as the F1 value [MSE06], which among other things evaluates the cluster purity. It is computed as the harmonic mean of recall ("are all clusters detected?") and precision ("are the clusters accurately/purely detected?"). Precision and recall are computed for each given class label, by comparing this set of labeled objects with all objects that have been detected in a subspace cluster showing this class label as the most frequent one among the clustered objects. Thus, F1 measure is able to compare results of different sizes. The F1 value of the entire clustering is the average of all class labels' F1 values. As a widely used measure, F1 is of key importance for comparability of experiment results. Further details about its characteristics in comparison with other measures are given in Chapter 10. In addition to the F1 measure, we provide the commonly used accuracy of classifiers

[1] http://dme.rwth-aachen.de/OpenSubspace/RESCU

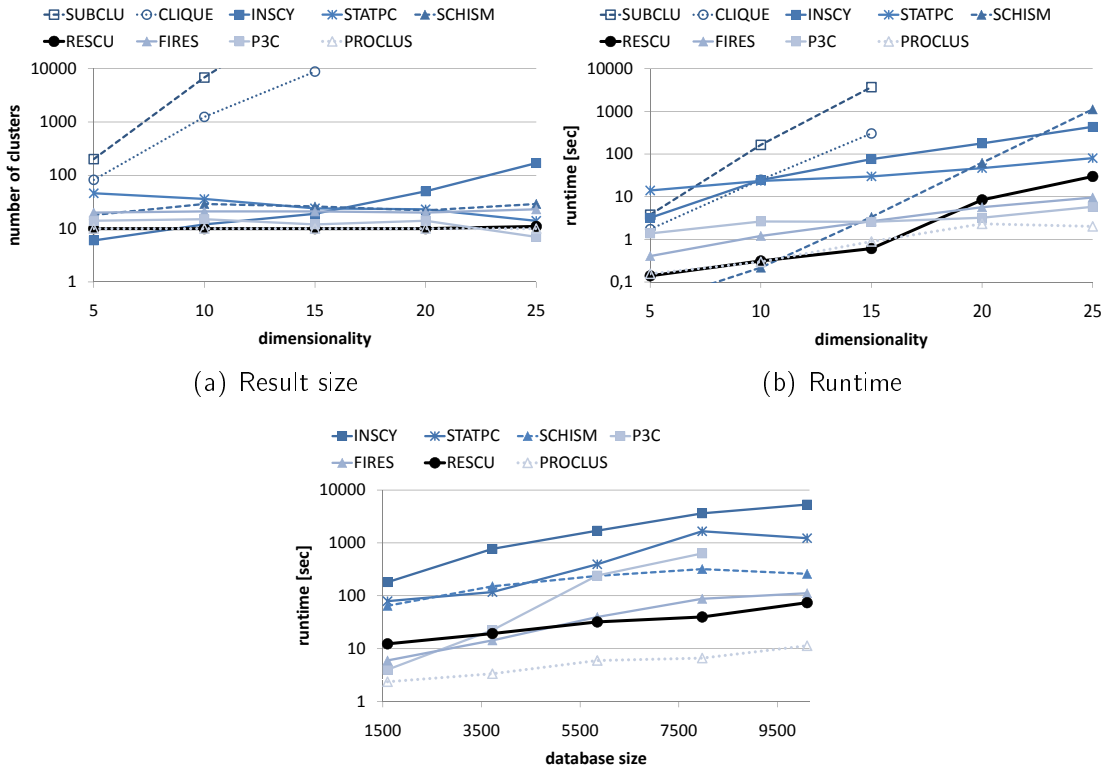
(e.g. C4.5 decision tree) built on the detected patterns [BZ07]. A high accuracy indicates that the subspace clustering is a good generalization of the underlying data distribution. For more details on these two measures please refer to our evaluation study in Chapter 10. As in our evaluation study we provide both measures on many data sets for the evaluation in this chapter, providing overall a thorough analysis for real world data.

3.4.1 Scalability

Our comparative study begins with an analysis of the result size and runtime (Fig. 3.5(a) & 3.5(b)) with respect to the dimensionality on a synthetic data set with 10 hidden clusters.

Redundancy maintenance: First, we want to show the result size of the algorithms CLIQUE and SUBCLU, which do not provide redundancy removal. They produce overwhelming result sets that are several orders of magnitude larger than the hidden clusters. They suffer from the fact, that any cluster typically induces several very similar clusters in lower dimensional projections, especially as dimensionality increases. Our RESCU approach successfully detects only the 10 relevant subspace clusters. Figure 3.5(b) illustrates the effect of size on the runtime of these approaches. (Please note the logarithmic scale.) Beyond 15 dimensions, the poor scalability of CLIQUE and SUBCLU due to their result size renders analysis infeasible on standard desktop PCs. In the following experiments, we do not include CLIQUE and SUBCLU due to their poor performance in terms of result size and runtimes. RESCU, as a representative for models with redundancy removal, clearly outperforms these algorithms. In summary, redundancy removal is a key property to provide interpretable results and good scalability. Algorithms such as CLIQUE and SUBCLU that keep redundant clusters in their output are not considered in the following experiments.

Redundancy removal: Compared to CLIQUE and SUBCLU the remaining algorithms scale to higher dimensional data sets and show significantly smaller result sizes in the range from 6 to 169 as depicted in Figure 3.5(a). Please note that comparison with some algorithms that require the number of clusters as an input, like the projected clustering approach PROCLUS, does not reflect their performance in real application scenarios where this information is typically not available. Most of the algorithms output more than 10 clusters, however, they were not able to detect all of the 10 hidden clusters. RESCU is able to find all hidden clusters. Overall only our RESCU relevance model is able to find the hidden and thus relevant clusters. We further investigate the quality of the clustering result on real world data sets in Section 3.4.2. The runtime of RESCU (Fig. 3.5(b)) is comparable to that of the other approaches. It is less affected by the dimensionality as it computes only the relevant subspace clusters on-demand and excludes most irrelevant results. Our relevance ranking of cluster candidates and the greedy processing ensures an overall



(a) Result size

(b) Runtime

(c) Runtime

Figure 3.5: Scalability w.r.t. dimensionality and database size

efficient computation. Although some algorithms have smaller runtime, RESCU is still efficient and, as we believe, this aspect is compensated by the higher clustering quality of RESCU.

Our next experiment in Figure 3.5(c) shows the runtime of the algorithms with respect to the database size. Our RESCU approach scales well whereas most of the competing algorithms show a greater increase in runtime. Overall the results from this experiment are comparable to the results in Fig. 3.5(b).

3.4.2 Quality on real world data

Our next experiment in Figure 3.6 evaluates F1 value and accuracy for the pendigits data. We vary the dimensionalities from 16 to 48. For F1 measure on the 16d data set we observe top quality results for RESCU, P3C and PROCLUS. Hence these algorithms find almost all and pure clusters. While P3C does not scale to the higher dimensional data sets, RESCU and PROCLUS reach again high qualities. Considering accuracy, our RESCU approach has the best quality results. SCHISM is second in accuracy, but has a significantly lower F1 value. In general, RESCU is the only approach that shows top results for both measures in all dimensionalities. Our novel model is able to detect the most interesting and non-redundant clusters in the data set.

It must be highlighted that the results of the two density-based approaches INSCY

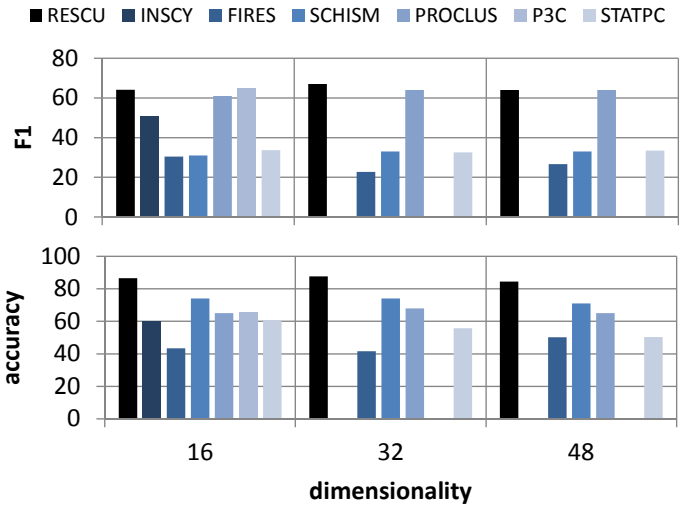


Figure 3.6: Quality on pendigits data (7494 objects; 16-48 dimensions)

and FIRES cannot compete with RESCU. The high quality of RESCU is not only a result of the density-based instantiation of the subspace cluster definition but in particular due to our new relevance model.

In Table 3.1 we show F1 and accuracy results for six further real world data sets. In addition to the absolute values we note the relative quality compared to the best measurement on each data set. Best 95% results are highlighted in gray. RESCU achieves top quality results for *all* data sets with respect to *both* measures. Competing approaches show highly varying performance. None of them achieves top quality allover. Although some of the approaches achieve slightly better results on some of the data sets, RESCU reliably shows top results on all data sets.

	Glass (214; 9)				*Vowel (990; 10)*				*Diabetes (768; 8)*			
	F1		*Accuracy*		*F1*		*Accuracy*		*F1*		*Accuracy*	
RESCU	60	100%	62	100%	44	100%	64	96%	71	100%	69	100%
INSCY	56	93%	54	87%	37	84%	67	100%	58	82%	65	94%
FIRES	30	50%	49	79%	10	23%	12	18%	33	46%	65	94%
SCHISM	45	75%	49	79%	24	55%	53	79%	69	97%	69	100%
PROCLUS	39	65%	54	87%	32	73%	30	45%	44	62%	65	94%
P3C	17	28%	39	63%	8	18%	16	24%	44	62%	65	94%
STATPC	19	32%	47	76%	17	39%	47	70%	39	55%	64	93%

	Shape (160; 17)				*Liver-Dis. (345; 6)*				*Breast (198; 33)*			
	F1		*Accuracy*		F1		*Accuracy*		*F1*		*Accuracy*	
RESCU	60	100%	75	100%	62	97%	61	98%	67	100%	76	97%
INSCY	56	93%	61	81%	62	97%	59	95%	65	97%	70	90%
FIRES	56	93%	62	83%	50	78%	53	85%	46	69%	75	96%
SCHISM	38	63%	59	79%	64	100%	58	94%	65	97%	71	91%
PROCLUS	60	100%	62	83%	46	72%	62	100%	47	70%	77	99%
P3C	39	65%	45	60%	36	56%	58	94%	63	94%	77	99%
STATPC	31	52%	62	83%	57	89%	58	94%	41	61%	78	100%

Table 3.1: F1 and accuracy measured for different real world databases [Captions: data set (size; dimensionality)]

3.4.3 Parametrization

We also evaluate the approximation quality of RESCU compared to the full optimization model. As we have proven RESCU is NP-hard and thus we can compute an optimal solution only for very small settings. In this experiment we use datasets with 10 clusters. The cost and objects per cluster are randomly selected from $(0; 1]$ and $[1; 100]$, respectively. For each Δ value we generate at least 20 random datasets to calculate the optimal and approximative solution. Figure 3.7 gives the relative approximation quality compared to the optimal solution (in terms of cost) for different values of Δ. On average, the approximative solution shows only small differences to the optimal solution. The whiskers, corresponding to the 15% and 85% quantiles respectively, indicate that also rare cases yield good approximation qualities. The approximation quality is remarkably robust against the parameter Δ, the cluster gain threshold. Even for extremely rigid values close to zero, average approximation is close to optimal.

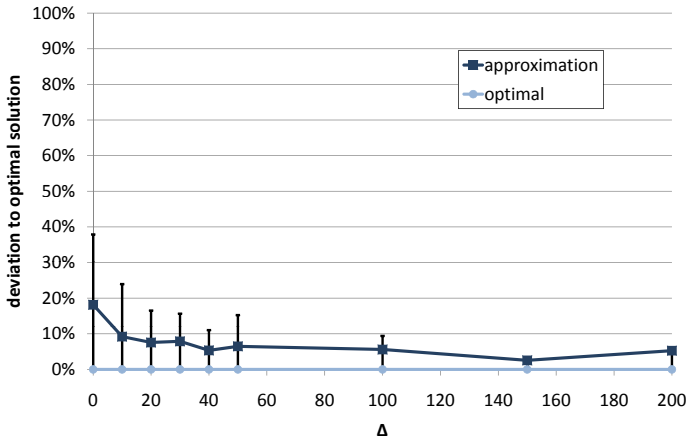


Figure 3.7: Approximation quality (Δ variation)

Next, we evaluate the flexibility of our RESCU model. For our cost function instantiation (cf. Sec. 3.2.5), we vary the β parameter that controls interestingness as a trade-off between higher dimensional clusters and more objects per cluster. For the 32d pendigits data set, Figure 3.8 shows the median size (triangles) of the clusters and the number of high-D (≥ 12) clusters (squares) for the relevant clustering of a given β value. As we can see, low β values give strong preference to large and few clusters, whereas high values result in many clusters with less objects. Likewise, RESCU can be just as easily adapted using any other cost function that reflect the interestingness in a user's analysis.

3.5 Enhancements due to the proposed optimization

We introduce the *RESCU* (relevant subspace clustering) model for mining the most interesting non-redundant clusters in high dimensional data. Our novel model incorporates both non-redundancy and global interestingness via a new cluster gain

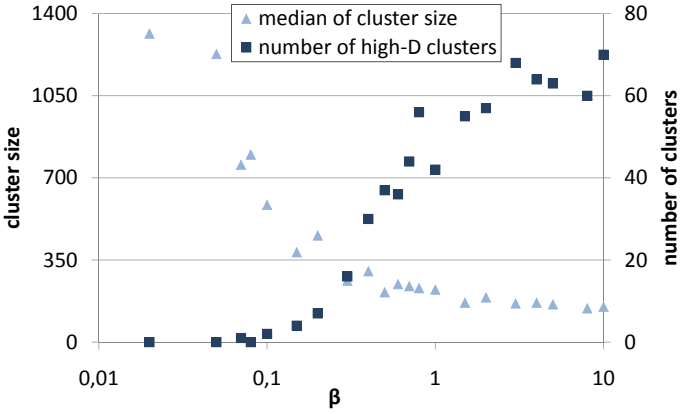


Figure 3.8: Effects on clustering by β variation

definition. Thus, we tackle Challenge 3 by including all and only the most relevant subspace clusters into the result set. Furthermore, we tackle Challenge 1 as we propose an adaptive density measure by varying directly the neighborhood range ε in the instantiation of our density-based subspace cluster definition. Our relevance model may also detect multiple concepts for some objects, but does not actively search for multiple hidden concepts (cf. Challenge 2). However, we tackle this challenge by an active search of multiple orthogonal concepts in the following chapter.

Overall, we prove that the computation of our relevance model is NP-hard which derives new efficiency challenges. For the computation of *RESCU*, we propose a relaxation of the relevance in our subspace clustering result that shows high accuracy with respect to our relevance model. Further developments considering the efficient computation of this model are described in Part II of this thesis. Thorough experiments on the clustering quality of our model demonstrate that *RESCU* reliably outperforms existing subspace and projected clustering algorithms while automatically reducing the output to all and only relevant clusters.

Chapter 4

Multiple concept detection in orthogonal subspaces

In today's applications, for each object very many attributes are provided, such that multiple concepts described by different attributes are mixed in the same database. In these high dimensional data spaces, each object can be clustered in several projections of the data. However, recent clustering techniques do not succeed in detection of these orthogonal concepts hidden in the data. They either miss multiple concepts for each object by partitioning approaches or provide redundant clusters in very similar subspaces. In our previous work we address multiple concepts by allowing objects to be detected in multiple subspace clusters. However, in the previous chapters we have not considered similarity of different subspaces for our redundancy definition. Even with our optimization approach, we consider redundancy only based on the object sets ignoring the similarity of their subspaces.

In this work we propose a novel clustering method aiming only at orthogonal concept detection in subspaces of the data. Unlike existing clustering approaches, OSCLU (Orthogonal Subspace CLUstering) detects for each object the orthogonal concepts described by differing attributes while pruning similar concepts. Thus, each detected cluster in an orthogonal subspace provides novel information about the hidden structure of the data. In contrast to clustering models described in the previous chapters, our orthogonal subspace clustering model takes similarity of subspaces into account. Thus, compared to our relevance model described in Chapter 3, we exclude further redundant clusters which are detected in similar subspaces while including novel knowledge in orthogonal subspaces. Thorough experiments on real and synthetic data show that OSCLU yields substantial quality improvements over existing clustering approaches aiming at detection of multiple concepts for each object.

4.1 Motivation of multiple concepts

In the knowledge discovering process, clustering aims at detecting groups of similar objects while separating dissimilar ones. Traditional clustering approaches compute a partition of the data, grouping each object in at most one cluster or detecting it as noise. However, it is not always the case that an object is part of only one cluster.

Multiple meaningful groupings might exist for each object. The detection of such multiple clusters describing different views on each object is still an open challenge in recent applications.

In today's applications, data is collected for multiple analysis tasks. In most cases, databases contain objects specified by very many attributes. As one does not know the hidden structure of the data, one mixes up different measurements in one high dimensional database. Thus, each object can participate in various groupings reflected in different subsets of the attributes. For example, in customer segmentation, objects are customers described by multiple attributes specifying their profile. A customer might be grouped by the attributes "average fruit consumption" and "sport activity" with other "healthy people" having high values in both of these attributes. The same customer might be a "Rock Fan" which could be specified by high values in the attribute "attendance to rock concerts" and low values in "attendance to classic concerts" (cf. Fig. 4.1). We observe for each customer multiple possible behaviors which should be detected as clusters. Thus, clusters may overlap in their clustered objects such that each object is represented by multiple clusters. Furthermore, each behavior of a customer is described by specific attributes. Thus, meaningful clusters appear only in these specific subspace projections of the data. While the attribute "attendance to rock concerts" is useful for the distinction of musical interests, the attribute "fruit consumption" is irrelevant for grouping musical interests of customers.

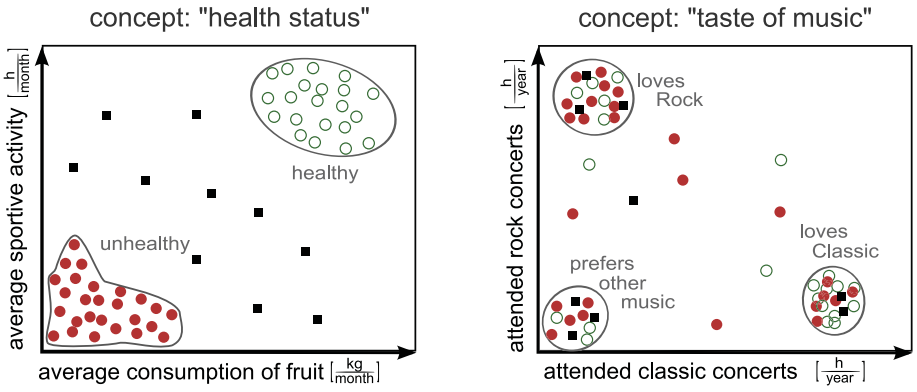


Figure 4.1: Example of different concepts

We generalize these observations as they are not only applicable to customer segmentation. In other applications, objects might be sensor nodes represented by multiple sensor measurements, or objects might be genes described by their expression level under multiple conditions. For each of these application scenarios, objects are described by very many attributes. For such high dimensional data, all objects seem to be unique in full space as distances grow alike due to the so called "curse of dimensionality". However, a common observation is that each of the objects might be part of different groups in different subsets of attributes. In general, we call this an object that is part of multiple *orthogonal concepts*. All of these groupings are valid characterizations of the same objects by using different attributes in *orthogonal subspaces*. Thus, for the general case of high dimensional data, a concept is

described by a subset of the dimensions. Hence we can substitute the detection of orthogonal concepts with the detection of orthogonal subspaces and their contained clusters. In our example, some healthy and unhealthy customers group together in another subspace and we can detect the orthogonal concept that represents the customer's taste of music. The same observation can be made in the other scenarios as well: Genes are controlling multiple functions (concepts) expressed only under specific conditions (relevant attributes for the concept), or sensors are measuring multiple concurrent environmental events (concepts) specified by different sensor measurements (relevant attributes).

Detection of Orthogonal Concepts Formally, a concept is an interpretation of a group of objects considering a set of relevant attributes (specific for this concept). The main characterization of a concept is given by its relevant dimensions as these dimensions provide the attributes in which the objects are grouped together. In Fig. 4.1 the concept "taste of music" is described by two dimensions. A concept can contain several groups that are clearly separated in the relevant dimensions of the concept, like customers loving Rock or customers loving Classic in our previous example. Each object may be clustered by at most one cluster in the same subset of relevant attributes. However, it can be clustered in multiple orthogonal concepts having different sets of attributes. Considering the concept "health status", a "Rock Fan" can be clustered with other customers to form a new grouping. There might exist multiple meaningful groups for each object as it can be interpreted in multiple different ways. Thus, orthogonal concepts provide for each object different groupings using no or only few shared attributes. Our novel OSCLU (Orthogonal Subspace CLUstering) approach detects for each object multiple orthogonal concepts. Each detected cluster provides novel information, as we aim at detection of only clusters in orthogonal subspaces. Thus, OSCLU prunes the detection of similar concepts by ignoring clusters in similar subspace projections of already detected clusters.

Summing up, in our approach we aim at detection of only the orthogonal concepts fulfilling the following properties:

- subspaces and subspace clusters represent the concepts in the database
- objects might be present in multiple clusters, if the subspaces of their concepts differ (to a high extent)
- each cluster provides novel information for its concept

Following these properties, we propose a method for selection of orthogonal subspaces by using a similarity measure on subspace projections. Our novel approach OSCLU chooses according to this similarity only the orthogonal subspace to include novel concepts in this subspace projection into the result set. In addition, we propose a relaxation of the orthogonal subspaces to "almost orthogonal subspaces". This generalization allows us to detect concepts sharing a certain amount of common dimensions. The attribute "gender" for example could belong to several concepts.

Relaxing to almost orthogonal subspaces includes more possible concepts in the result.

As each object might be present in multiple clusters, we have to ensure that each cluster adds sufficiently novel information within its concept. Unlike most subspace clustering techniques we prevent redundant information. For this purpose we introduce an interestingness measure for choosing only sufficiently distinguishing clusters from similar concepts. Furthermore, to select the most interesting clusters, we present an objective function that is based on multiple properties like size, dimensionality and density extracted out of the subspace clusters.
Using both properties of orthogonal subspaces and most interesting clusters, OSCLU performs a global optimization of the result set. It ensures to include overlapping clusters to detect multiple concepts. Furthermore, it prunes similar subspaces and non-interesting clusters to ensure only the meaningful patterns in the result.

4.2 Comparison with related work

Different clustering paradigms have been proposed in the past decades. In this section, we review the main techniques and show their drawbacks in detection of orthogonal concepts in high dimensional data. Especially, we show the differences to our approach as none of the proposed techniques analyzes subspace projections to steer cluster detection in the direction of orthogonal concepts.

Traditional Clustering Traditional clustering approaches aim at the detection of clustered objects using all attributes in the full data space. Independent of the underlying clustering model, full space clustering approaches are unable to detect multiple orthogonal concepts, as these clusters do not appear across all attributes.

Clustering in Subspace Projections Clustering in subspace projections aims at detecting clusters in arbitrary subspaces. While *projected clustering* techniques detect disjoint subsets of objects [AWY+99, MSE06], these approaches miss to detect orthogonal concepts. They are unable to detect multiple clusters per object. In contrast, *subspace clustering* allows objects to be part of multiple clusters in arbitrary subspaces. However, approaches either miss to remove redundancy [AGGR98, KKK04] or as proposed by our own approaches reduce redundancy without considering similarity of subspaces. However, the similarity of subspaces is crucial for orthogonal concept detection. In contrast to clustering models described in the previous chapters, our orthogonal subspace clustering model takes similarity of subspaces into account. Thus, compared to our relevance model described in Chapter 3, we exclude further redundant clusters which are detected in similar subspaces while including novel knowledge in orthogonal subspaces.

Orthogonal Clustering Recent extensions of traditional clustering techniques try to iteratively detect further orthogonal multi-view clusters [CFD07, QD09]. In each

step, these approaches transform the data space and thus force the traditional clustering algorithm to find novel clusters in orthogonalized spaces. In contrast to subspace clustering algorithms, they search for clusters in full space or in space transformations. By transforming the data they pose constraints on the cluster detection. However, transformation hinders the interpretation of results as the original attributes are not directly used for description of detected concepts. Furthermore, due to the iterative procedure, orthogonal clustering typically detects some of the already detected clusters multiple times [CFD07] or is restricted to one alternative clustering for a given set of clusters [QD09]. Detection of only orthogonal concepts is hindered as redundant information does not provide any additional knowledge. Furthermore, restriction to a fixed number of alternative concepts does not provide knowledge about all hidden concepts.

Multi-Source Clustering In contrast to previous approaches, multi-source clustering does not tackle the challenge of concept detection. One simply assumes to have the knowledge about the concepts provided by different sources [BS04]. Approaches in this paradigm try to detect clusters in the given multi-source data which already provide the different views on the data. Similarly, approaches in the research areas of parallel universes and relational data mining [WHB10, Dv03], assume also multiple given data sources. Thus, in contrast to our approach, they all assume they know about the relevant dimensions for each concept. In general, this is not true for orthogonal concept detection as multiple concepts can be hidden in one high dimensional data source. Thus, multi-source clustering techniques are not able to detect multiple concepts in a single data source. Furthermore, they are also not able to detect concepts spreading across two or more data sources computing new subsets of relevant dimensions.

4.3 Orthogonal concepts in subspaces

In this section, we present our model for the detection of orthogonal concepts in subspaces of high dimensional data. Formally we map our contributions to an optimization problem based on detected subspace clusters in the database. In contrast to subspace clustering, where all clusters are selected for the result set, we choose only a subset of most interesting clusters based on orthogonal subspaces. As in the previous chapter, we make a distinction between the cluster definition and clustering definition. While the cluster model defines the properties that a set of objects $O \subseteq DB$ and a set of dimensions $S \subseteq Dim$ have to fulfill to be a valid cluster $C = (O, S)$, the clustering model determines a set of clusters $M = \{C_1, \ldots, C_n\}$ to be a valid clustering. The valid clustering for traditional subspace clustering is simply the set ALL that contains all subspace clusters. As this set is highly redundant, we do not consider it a valid clustering in our model.

We want to generate a most informative clustering $OPT \subseteq ALL$ so that the clusters in the result set represent the multiple concepts of the data without obfuscating this structure by redundant information. As motivated before, each object

might be present in multiple clusters if the clusters describe different concepts and each cluster $C \in OPT$ has to provide novel information within its similar concepts. In short it is not allowed to group the same objects in similar concepts by several clusters. Therefore, we have to define

- whether a concept is similar to another one or if it describes a different concept
- and if a cluster identifies a new grouping within its similar concepts.

As a consequence overlapping clusters between different concepts are possible, in contrast to projected clustering. We solely have to check if the same objects (overlapping objects) are already described within similar concepts (overlapping dimensions) to filter out uninteresting clusters and to steer our cluster detection to the orthogonal subspaces. For a flexible model we introduce two main parameters α and β controlling the intended overlap in objects and dimensions respectively. In a first step in Section 4.3.1 we define the notion of (almost) orthogonal concepts, to determine which concepts are similar to a selected one. In Section 4.3.2 we present the interestingness criterion that each cluster has to fulfill to be an informative cluster within its similar concepts. In Section 4.3.3 we define our overall model for the optimal orthogonal clustering and show in Section 4.3.4 how the user can influence the clustering result. In Section 4.3.5 we prove that solving this model is NP-hard.

4.3.1 Definition of almost orthogonal concepts

The data collected in today's applications are generated by different concepts which are mixed together. In an optimal setting the concepts, described by subspaces, share no dimensions and we can clearly distinguish between them. If we identify a concept in the subspace S, all other subspaces T which share at least some dimensions $T \cap S \neq \emptyset$ are similar to it and we can prune them. T cannot characterize a different concept because a dimension $d \in S \cap T$ is already covered by the concept in S and hence T does not detect a novel concept in this scenario. Hence, all subspaces that are similar to S are excluded from further consideration by the identification of S. This can be formalized by:

$$\begin{aligned} coveredSubspaces_0(S) &= \{T \subseteq Dim \mid T \cap S \neq \emptyset\} \\ &= \{T \subseteq Dim \mid |T \cap S| > 0\} \end{aligned}$$

A concept with the relevant subspace T is orthogonal to a concept in S if $T \notin coveredSubspaces_0(S)$. The dimensions of T and S are disjoint and hence we can detect novel information in T. So our clustering model only has to identify clusters in subspaces which are orthogonal and prune the already covered subspaces.

However, this orthogonality definition is a too hard restriction for our clustering model. Many subspaces are prohibited for selection and hence the resulting clustering contains only low information. By definition, each dimension appears in at most one concept. However, overlapping concepts are useful and expected in real life scenarios, e.g. the attribute "gender" in a customer database could appear in multiple concepts. For subspace clustering we need a relaxation of the orthogonality property.

A less hard restriction is realized by the idea of excluding lower dimensional projections of S. The subspace S is more meaningful for the representation of a concept than using the projections which contain fewer attributes. Hence, if we identify S as the relevant subspace for a concept each projection is already described by this subspace. The subspaces similar to S can be defined by:

$$\begin{aligned} coveredSubspaces_1(S) &= \{T \subseteq Dim \mid T \subseteq S\} \\ &= \{T \subseteq Dim \mid T \cap S = T\} \\ &= \{T \subseteq Dim \mid |T \cap S| = |T|\} \end{aligned}$$

By this definition we can find overlapping concepts, e.g. characterized by $S_1 = \{1, 2\}$ and $S_2 = \{2, 3\}$. Neither of them is similar to the other concept and hence both of them could appear in the result set. This definition is related to the maximality property in our previous subspace clustering approaches (cf. Chapter 2), resulting in the same problems. Even if two subspaces share a high fraction of dimensions, e.g. 9 out of 10, they represent different concepts. Thus, similarity of subspaces is not yet modeled in an adequate way.

Our model of almost orthogonal concepts integrates the advantages of both models. We allow overlapping concepts, but we also avoid concepts with too many shared dimensions. Thus, we only include (almost) orthogonal concepts in the result and obtain a flexible model by generalizing both definitions to:

$$coveredSubspaces_\beta(S) = \{T \subseteq Dim \mid |T \cap S| \geq \beta \cdot |T|\}$$

with $0 < \beta \leq 1$. For $\beta \to 0$ we get the first, for $\beta = 1$ the second definition. As one of our main parameters for a flexible clustering model, β allows to specify the amount of overlap in terms of dimensions. Allowing more overlap by increasing β we allow clusters to be detected in some similar subspaces, while decreasing β to 0 prohibits overlap in dimensions such that each dimension is used only once (partitioning of dimensions).

The idea of our clustering model is to avoid the grouping of the same objects in similar concepts by several clusters. Given a cluster C we have to determine the set of clusters that are in similar concepts. Because we use orthogonal subspaces for the orthogonal concept detection, we can determine these clusters by checking if their subspaces cover the subspace of C. We call this set the concept group of C which can be formalized by the following definition.

Definition 15. Concept group
The concept group of $C = (O, S)$ with respect to a set of clusters $M = \{C_1, \ldots, C_n\}$ is defined as

$$conceptGroup(C, M) = \{C_i \in M \backslash \{C\} \mid S \in coveredSubspaces_\beta(S_i)\}$$

The concept group of $C = (O, S)$ contains all clusters that share at least a β-fraction of the dimensions of S. Checking the grouped objects O of C against the objects of its concept group is required to provide novel information within similar

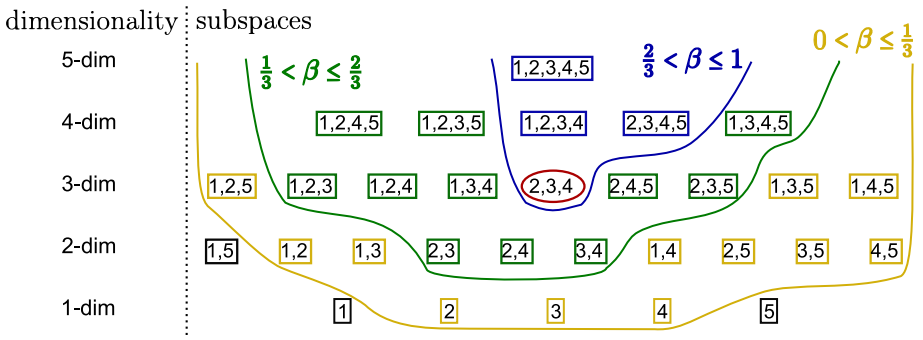


Figure 4.2: Concept group with variation of β

concepts. All other clusters, not in the concept group of C, do not need to be considered because they belong to other concepts. We permit such multiple concepts in our result.

Let us consider Figure 4.2 where the selected cluster C is in the subspace $\{2, 3, 4\}$. For $\beta \to 0$ we have to compare C with all clusters in subspaces sharing at least one dimension. C has to group new objects w.r.t. these clusters because they all characterize similar concepts. The higher β the less subspaces are considered as similar and hence the more concepts are possible in the final clustering. The choice of $\beta = 1$ results in comparing C only to higher dimensional clusters C', which project to the subspace of C. For example the concept described by the subspace $\{1, 2, 3, 4\}$ subsumes the concept of C and thus C has to be checked against this subspace. Thereby, we see that the concept group is not symmetric but it tends to include more higher dimensional clusters. The concept group of a low dimensional cluster, that is in general less interesting, usually contains more clusters compared to the one of a higher dimensional cluster. Thus, for a low dimensional cluster it is more difficult to provide novel information and consequently to be included in the result set.

4.3.2 Global interestingness of the overall clustering

After defining the clusters which characterize similar concepts as C, we have to ensure that the cluster is interesting enough compared to these clusters. For our resulting clustering $OPT \subseteq ALL$, each cluster $C \in OPT$ has to fulfill this property. According to our motivation a cluster $C = (O, S)$ has to group new objects within the similar concepts. Hence we use the coverage of objects as a criterion for interestingness. For a clustering $M = \{C_1, \ldots, C_n\}$ the coverage is defined as:

$$Coverage(M) = \bigcup_{i=1}^{n} O_i$$

Because a strict partitioning of the clusters in similar concepts is not useful, i.e. we would enforce that each object of C is in no other cluster, we relax this property. We calculate the relative fraction of objects which are not covered by other clusters in similar concepts w.r.t. the whole cluster size. In contrast to our cluster gain proposed

in Definition 10, our global interestingness takes similarity between subspaces into account. As only objects covered by clusters in the concept group are excluded, this definition assigns low interest to similar subspace clusters while orthogonal concepts show up with high global interest.

Definition 16. Global interestingness
Given a cluster $C = (O, S)$ and a set of clusters $M = \{C_1, \ldots, C_n\}$. The global interestingness of C with respect to M is

$$I_{global}(C, M) = \frac{|O \backslash Coverage(conceptGroup(C, M))|}{|O|}$$

First, we determine the clusters in similar concepts to the one of C and afterwards their objects are removed from O to obtain the newly covered objects of C. Only if $I_{global}(C, M)$ is larger than a given threshold α the cluster adds sufficiently new information to this concept.

Figure 4.3 illustrates this interestingness check. Let us assume that M contains the clusters C_7 to C_{10} and possible further clusters in other subspaces (not within dimension 3). If we choose $C = C_{10}$ the concept group corresponds to $\{C_7, C_8, C_9\}$. The remaining clusters are not considered because they represent other concepts. C_{10} has to group new objects within the concept. However, most of the objects (29 out of 32) from C_{10} are already covered by the other clusters and hence the information obtained by C_{10} in this concept is small ($I_{global}(C, M) = \frac{32-29}{32}$). For a threshold $\alpha > \frac{3}{32}$ the cluster C_{10} is regarded as redundant with respect to M.

The user is able to control the required interestingness of a cluster by variation of α. If the fraction of newly clustered objects is smaller than α we do not choose the cluster. For the extremal value $\alpha = 1$ the clusters in similar concepts must not overlap (partitioning of objects per concept group). For $\alpha \to 0$ a cluster is selected as long as not all objects are covered by other clusters. Consequently a high overlap is possible. Overall, our main parameters α (object overlap) and β (dimension overlap) are able to provide a high flexibility of setting preferences in our clustering model.

An important aspect of this model is that the interestingness is checked against several clusters within similar concepts. Unlike our first model that makes only a pairwise comparison of clusters, in our novel model all clusters from a similar concept are considered at the same time to evaluate the interestingness of the new cluster. If we did not check against several clusters, the cluster C_{10} in Figure 4.3 would get a misleading high interestingness value. A pairwise comparison of C_{10} to C_7 or C_8 indicates a high fraction of newly clustered objects which is in fact not true.

Let us choose a clustering $M \subseteq ALL$. The global interestingness ensures that each cluster $C \in M$ results in an information gain within its concept by covering new objects. Varying concepts are possible in M and considered by the definition. Thus, the proposed properties for a good clustering, mentioned at the beginning of the section, are guaranteed.

Definition 17. Orthogonal clustering
The clustering $M = \{C_1, \ldots, C_n\}$ is orthogonal iff

$$\forall C \in M : I_{global}(C, M \backslash \{C\}) \geq \alpha$$

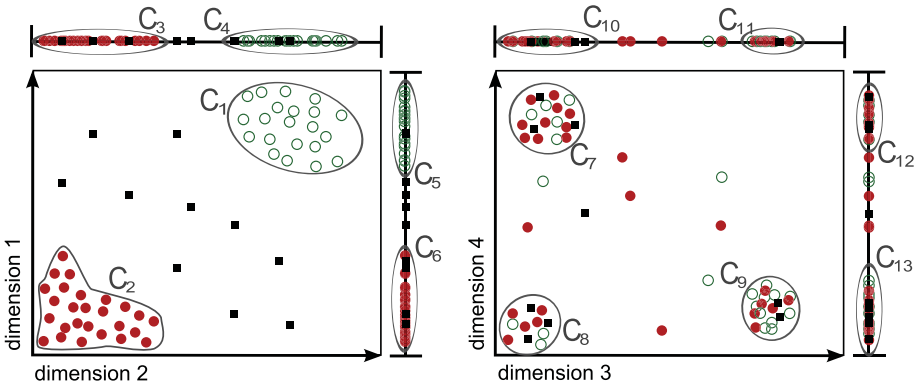


Figure 4.3: Example for the selection of an optimal orthogonal clustering

The clustering $M = \{C_1, C_2, C_7, C_8, C_9\}$ in our example from Fig. 4.3 is an orthogonal clustering, while the clustering $M \cup \{C_{10}\}$ is not. However, the proposed definition alone is not yet sufficient to determine an optimal clustering $OPT \subseteq ALL$. Several clusterings could fulfill the definition, e.g. the trivial clustering $M = \emptyset$. The user wants to get an overview of the clustering structure and seeks for the most informative clusters. We have to ensure that these clusters are selected.

4.3.3 Optimal orthogonal clustering

While the global interestingness $I_{global}(C, M)$ always rates the cluster C with respect to a clustering M we now assess the interestingness of the cluster C on its own. This so called local interestingness should correspond to the user-specific notion of interesting clusters. Formally we have to define a function I_{local} which maps each cluster C to the value $I_{local}(C)$. This function could include different aspects, as the dimensionality or the size of the clusters. A discussion of this function is presented in Section 4.3.4.

Both, the global and local interestingness are used to define our optimal orthogonal clustering. With the global property we ensure that only informative clusters within similar concepts are selected. At the same time we want to maximize the sum of the local interestingness for the resulting clusters. By maximizing the local interestingness we get the most interesting clusters but also as many interesting clusters as possible (taking the orthogonal clustering constraint into account).

Definition 18. Optimal orthogonal clustering (OOC)
Given the set ALL of all possible subspace clusters, a clustering $OPT \subseteq ALL$ is an optimal orthogonal clustering iff

$$OPT = \arg \max_{M \in Ortho} \left\{ \sum_{C \in M} I_{local}(C) \right\}$$

with

$$Ortho = \{M \subseteq ALL \mid M \text{ is an orthogonal clustering}\}$$

In Figure 4.3 we show an overall example with $\alpha = 0.5$ and $\beta = 0.5$. The clustering $M_1 = \{C_1, C_2, C_7, C_8, C_9\}$ is a valid orthogonal clustering, because each cluster covers a sufficient amount of new objects within its concept. Although C_1 and C_9 contain similar objects the overlap is permitted because different concepts are realized. The clustering $M_1 \cup \{C_{10}\}$ for example is not valid because as shown in our previous example $I_{global}(C_{10}, M_1) = \frac{32-29}{32} < \alpha$. Obviously, each subset of M_1 is also an orthogonal clustering but less informative than M_1. Hence these subsets cannot be optimal clusterings. If we assume that the user is more interested in high dimensional clusters and chooses I_{local} accordingly, the sum $\sum_{C \in M_1} I_{local}(C)$ would be maximal out of all orthogonal clusterings. Another orthogonal clustering like $M_2 = \{C_1, C_2, C_{10}, \ldots, C_{13}\}$ which contains the one-dimensional projections from the second concept would therefore result in a lower value for the sum. As a consequence, M_1 is preferred over M_2 and M_1 is the optimal clustering in this example.

Our model provides a selection of only interesting clusters in different and novel concepts. An overwhelming result size is prevented. As we use subspace clusters in our model, the interpretabiliy of the result set and the identification of the relevant attributes for each concept are guaranteed. Unlike other orthogonal clustering models we keep the original dimensions and we use them for the orthogonality check. We steer the cluster selection to orthogonal subspaces.

4.3.4 Local interestingness of one cluster

As in our previous work, we base on the density-based clustering paradigm. We use our adaptive density measure $density^S(p)$, which computes for an object p in subspace S the number of objects in its ε-neighborhood. As proposed in Section 3.2.5 we use a variable ε-neighborhood for comparable density measurements even in high dimensional subspaces. Based on this cluster definition, we can define our user-specific local interestingness function. In addition to our local interestingness defined in Section 3.2.5, we propose a more general interestingness for our OSCLU approach. We use three main properties to characterize a subspace cluster $C = (O, S)$ in this cluster instantiation. The dimensionality $|S|$, the size $|O|$ and the density. A very dense cluster shows small variation in the attribute values of the relevant dimensions and hence is more interesting than a sparse cluster. We use the mean density $\frac{1}{|O|} \sum_{p \in O} density^S(p)$ over all objects within the cluster for this criterion.

Maximizing all measures at the same time is in general not possible, e.g. low dimensional clusters are usually larger than high dimensional clusters. Therefore our local interestingness function subsumes all measures and gives the user the flexibility to weight the measures dependent on the application. The local interestingness function used in our experiments is

$$I_{local}(C) = |S|^a \cdot |O|^b \cdot \left(\frac{1}{|O|} \sum_{p \in O} density^S(p) \right)^c$$

with $C = (O, S)$ and $a + b + c = 1$.

4.3.5 Proof of NP-hardness

In this section we prove the NP-hardness of our optimal orthogonal clustering problem (OOC). For this we reduce the NP-complete $SetPacking$ problem [GJ79] to our model, i.e. $SetPacking \leq_P OOC$. Given several finite sets O_i the $SetPacking$ problem seeks for the maximal number of disjoint sets.

Theorem 2. *Computing OOC (Def. 18) is NP-hard.*

Proof.
We show that $SetPacking \leq_p OOC$.

A. Input mapping: Each set O_i is mapped to the cluster $C_i = (O_i, \{1\})$. Furthermore we set $\beta \in [0, \ldots, 1]$, $\alpha = 1$ and $I_{local}(C) = |S|$ (cf. Section 4.3.4, $a = 1, b = c = 0$).

B. OOC **generates a valid** $SetPacking$ **solution:**

1) The concept group contains all clusters:
$conceptGroup(C, M\backslash\{C\})$
$= \{C_i \in M\backslash\{C\} \mid S \in coveredSubspaces_\beta(S_i)\}$
$= \{C_i \in M\backslash\{C\} \mid \{1\} \in coveredSubspaces_\beta(\{1\})\}$
$= M\backslash\{C\}$

2) Each orth. clustering M contains only disjoint sets:
M is orthogonal clustering
$\Leftrightarrow \forall C \in M : I_{global}(C, M\backslash\{C\}) \geq 1$
$\Leftrightarrow \forall C \in M : \frac{|O\backslash Coverage(conceptGroup(C, M\backslash\{C\}))|}{|O|} \geq 1$
$\Leftrightarrow \forall C \in M : |O\backslash Coverage(M\backslash\{C\})| \geq |O|$
$\Leftrightarrow \forall C \in M : O \cap \bigcup_{C_i \in M\backslash\{C\}} O_i = \emptyset$

3) OPT contains maximal number of such disjoint sets:
$OPT = \arg\max_{M \in Ortho}\{\sum_{C \in M} I_{local}(C)\}$
$\Leftrightarrow OPT = \arg\max_{M \in Ortho}\{\sum_{C \in M} |\{1\}|\}$
$\Leftrightarrow OPT = \arg\max_{M \in Ortho}\{\sum_{C \in M} 1\}$
$\Leftrightarrow OPT = \arg\max_{M \in Ortho}\{|M|\}$

(2) and (3) $\Rightarrow$ OPT is a valid $SetPacking$ solution
$\Rightarrow$ OOC is NP-hard □

4.4 Computation of our complex model

The optimal orthogonal clustering has global properties which increases the computational complexity. As we have already proven the problem is NP-hard and hence we cannot expect that an efficient algorithm exists. Furthermore, we cannot generate the huge set of all subspace clusters ALL in a first step and select the optimal subset afterwards. We develop an approximation algorithm (OSCLU) that incrementally adds further clusters to the result set. In contrast to our previous solutions for optimizing the overall clustering result (cf. Chapter 3), for OSCLU we may use similarity

of subspaces for pruning the search space. For efficient calculation we integrate the clustering process into the concept and cluster selection process. This means that not all clusters in all subspaces are generated but many subspaces are pruned based on already detected clusters in similar subspaces. An important question is which subspaces should be clustered first and hence which clusters should be added at the beginning to the result set to prune many other subspaces.

Traditional bottom-up approaches that start with the low dimensional clusters are not useful for pruning based on our global interestingness criterion. As already mentioned, the concept group of a cluster contains mainly higher dimensional clusters (cf. Fig. 4.2). Thus, a low dimensional cluster has to compare its object coverage against more clusters than a high dimensional cluster. A low dimensional cluster is more likely to be excluded from the result set than a high dimensional cluster. For this reason, we use a top-down approach to add clusters to the final clustering.

Our algorithm, summarized in Algorithm 1, comprises three major contributions to avoid clustering of all subspaces. First we develop a ranking of the subspaces (all with the same dimensionality) without clustering them (lines 4-6). The ranking accounts for the similarity of the current subspace with already detected concepts. The greater the number of already detected similar concepts, the less interesting is the subspace. In a second step the ranking considers the possibility for a good clustering in a subspace based on efficient estimation. After ranking the subspaces we use the first subspace for clustering (line 8). If clusters were identified we incrementally update the result set (line 10). We have to consider the global interestingness so that redundant clusters are not selected. Furthermore, a high local interestingness of the selected clusters should be ensured. Resorting the ranking and the possible pruning of further subspaces (line 11) based on the new clusters is performed in order to push novel concepts to the top. If all subspaces with the dimensionality dim are pruned or selected for clustering we decrease the dimensionality to realize the top-down approach.

Algorithm 1 OSCLU (Orthogonal Subspace CLUstering)

```
 1: result set M := ∅;
 2: find initial dimensionality dim;                    (Sec. 4.4.3)
 3: WHILE(dim > 0)
 4:     rank and prune subspaces based on               (Sec. 4.4.1)
 5:         1) subspace orthogonality score
 6:         2) subspace quality score
 7:     WHILE(ranking not empty)
 8:         choose best subspace for clustering;
 9:         IF(clusters found)
10:             update result set M;                    (Sec. 4.4.2)
11:             resort ranking and prune;
12:     dim=dim-1;
13: return result set M;
```

As an additional step we present an efficient method that approximately identifies

the highest dimensionality (of a subspace) in which clusters are expected (line 2). This avoids to start our ranking in the full-space, where clusters are in general not present.

4.4.1 Orthogonal subspace selection

Clustering each subspace is not efficient since many subspaces can be pruned because of similar concepts that are already detected. We use two techniques to rank subspaces without clustering. The aim is to cluster only interesting and orthogonal subspaces. In our first approach we use the similarity of already discovered concepts for pruning and ranking. The greater the number of similar subspaces in the result set the higher is the possibility that new clusters in the current subspace cover the same concept and hence provide no novel information. We define the orthogonality score of a subspace S w.r.t. the current result set M as

$$orthogonality\_score(S, M) = \\ |\{T \subseteq Dim \mid S \in coveredSubspaces_{\beta}(T) \wedge \exists (O, T) \in M\}|$$

The definition is similar to the concept group, but only considers the subspaces and not the objects. The higher the score the worse is a subspace because many similar concepts are already in the result set. The orthogonality score is the first criterion for ranking. Furthermore, all subspaces with a score greater than $maxOrth$ are removed from the ranking. This parameter can be controlled by the user and intuitively defines how detailed a concept is analyzed.

During the algorithm the result set M changes and hence the orthogonality score does so too. By this procedure only the most informative subspaces are top ranked and hence are clustered. The clustering is concentrated to the orthogonal and novel concepts.

Our second approach makes use of subspace search [CFZ99, KKKW03] for measuring the quality of subspaces. Usually subspace search is a stand-alone technique for identifying interesting subspaces. Each subspace is mapped to a quality value, a high value corresponds to a high possibility for a good clustering structure. Our trick is to use this technique within the clustering task. We guide our algorithm to cluster only the most interesting subspaces based on the calculated qualities. Therefore our ranking is extended such that all subspaces with the same orthogonality score are secondly ranked based on this qualities. In total, our ranking concentrates not only on novel concepts but also on high quality subspaces.

The subspace search method within our framework is easily exchangeable and we can use techniques like RIS [KKKW03] or ENCLUS [CFZ99]. For efficiency reasons we develop an own technique which is based on our density-based cluster definition (cf. Sec. 4.3.4). To have clusters in a subspace several objects must have a high density according to our density-based clustering model, i.e. for an object p the value of $density^S(p)$ is large. We use a strategy that randomly selects points and calculates their mean density. This method is efficient and a good indicator for the

existence of clusters. Let $Seeds$ be the set of randomly selected points, the quality score is then defined as

$$quality\_score(S, M) = \frac{1}{|Seeds|} \sum_{p \in Seeds} density^S(p)$$

The higher the quality the better the subspace. As for the orthogonality score we introduce a minimum score $minQual$ that each subspace has to fulfill to be maintained.

4.4.2 Incremental result construction

After ranking the subspaces based on the two scores, we select the first one and we cluster it according to our model. We get a list of resulting clusters New. We now have to check which clusters $C \in New$ should be included to our result set M. In a first step we analyze the global interestingness of the new clusters. For each cluster $C \in New$ we calculate $I_{global}(C, M)$. We distinguish two cases.

If $I_{global}(C, M) \geq \alpha$, we directly add the cluster to M, i.e. the new result set is $M := M \cup \{C\}$. The cluster C adds sufficiently new information. By this procedure we ensure that in each step of the algorithm only informative clusters are selected. Please note that this procedure is a relaxation of Def. 17. We do not check the global interestingness of the remaining clusters in M which could be changed by selection of C. This recalculation would be too costly. However, due to our top-down approach higher dimensional clusters are added first to M, and these clusters are rarely removed by low dimensional clusters. Additionally, within the same dimensionalities, our ranking tries to rank the best subspaces on top and hence these clusters are selected first.

If $I_{global}(C, M) < \alpha$, we do not reject the cluster immediately but we perform an additional improvement heuristic. We want to maximize the local interestingness in our model hence we check if it is possible to remove some clusters from M such that C is afterwards globally interesting and the sum of the local interestingness is increased. The algorithm which decides if C is included and which subset of M should be removed is presented in Alg. 2. First, we rank the clusters from $conceptGroup(C, M)$ in decreasing order based on their local interestingness values. Second, we select the most interesting clusters which do not result in redundancy for C (set N). The clusters which induce the redundancy are stored in R. At the end it holds that $I_{global}(C, M \backslash R) \geq \alpha$, i.e. C provides novel information with respect to the new set. If the local interestingness of C is greater than the one of R it is better (for maximizing the interestingness) to select C and remove R from the result set M. The new result set is then $M := (M \backslash R) \cup \{C\}$. Otherwise C is rejected and the set M remains unchanged.

By the incremental result construction we add only informative clusters to our set and additionally try to maximize the interestingness of all selected clusters.

Algorithm 2 Cluster selection procedure

1: $< C_1, \ldots, C_n >$:= ranking of $conceptGroup(C, M)$
2: $N := \emptyset$, $R := \emptyset$
3: **FOR**(i:1 . . . n)
4: **IF**($I_{global}(C, N \cup \{C_i\}) \geq \alpha$) $N := N \cup \{C_i\}$
5: **ELSE** $R := R \cup \{C_i\}$
6: **IF**($I_{local}(C) > \sum_{C' \in R} I_{local}(C')$)
7: add C to M and remove R from M

4.4.3 Efficient initialization

In general, full dimensional clusters are not identified in high dimensional databases. If we started our top-down approach in full space we would analyze many uninteresting subspaces which are filtered out by our quality score criterion. For an efficiency boost we identify the first layer with interesting subspaces based on the idea of binary search. We start in the half-dimensional space (e.g. 5 in a 9d database) and use our subspace search estimator to calculate the qualities. If we identify a subspace with sufficiently high quality we directly continue with the dimensionality between half and full space (e.g. from 5 to 7). If no interesting subspaces are found we accordingly we continue with the lower dimensional spaces (e.g. 5 to 3). For this new dimensionality we repeat the binary search procedure (with half of the previous range) until we identify the highest dimensionality with interesting subspaces.

Overall, our algorithm comprises three contributions to obtain a good approximation of the optimal orthogonal clustering. The binary search technique supports the top-down approach by an efficient initialization. The ranking of subspaces yields a preference of orthogonal and interesting subspaces. By recalculating the ranking further subspaces can be pruned without clustering and novel concepts advance to the top. At last, the meaningful selection of new clusters to M results in an informative clustering. In the next section we confirm this with an experimental analysis.

4.5 Experiments

We evaluate the quality and efficiency of the *OSCLU* approach compared to three variants of orthogonal clustering techniques (*Multi-View 1* and *Multi-View 2* proposed in [CFD07], and *Altern. Clus.* [QD09]), a recent non-redundant subspace clustering technique (*StatPC* [MS08]) and a projected clustering approach (*P3C* [MSE06]). For fair comparison we used our evaluation framework, additionally reimplemented both Multi-View approaches in this framework and used the original implementation for the alternative clustering [QD09]. Furthermore, for all algorithms we tried to find the optimal parameter settings for each data set.

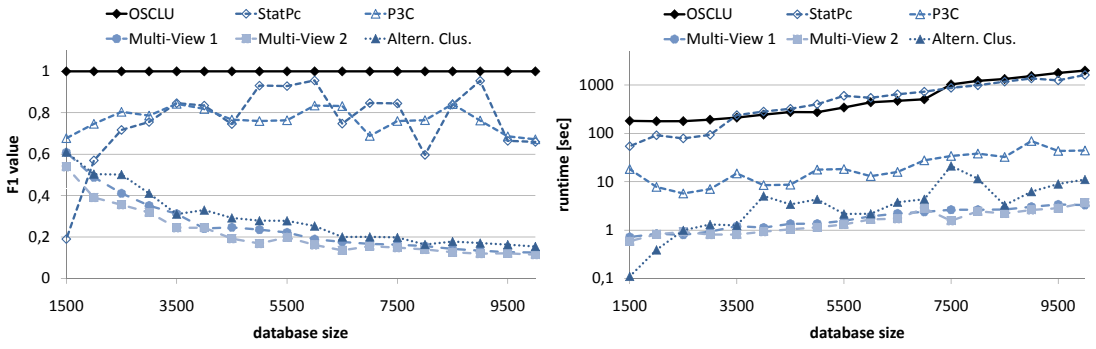
In general, we perform our evaluation on data with multiple hidden concepts. For both synthetic and real world data, we extend single concept data used in traditional clustering approaches such that each object is part of multiple concepts. Thus, for a high quality clustering each object has to be detected in multiple clusters. While tra-

ditional clustering approaches are well suited for data with only one hidden concept, we compare our approach against recent techniques designed for multiple hidden concepts. For scalability experiments, we generate synthetic data following a method proposed in [KKK04] to generate density-based clusters in arbitrary subspaces. In addition, our generator takes into account that objects can belong to multiple concepts. Thus, for each object we concatenated attribute values of different subspace clusters to a higher dimensional space with multiple hidden concepts per object. Further on, we show the performance of OSCLU on two extended real world data sets (original *iris* and *liver disorders* are provided by the UCI repository [AN07]). We use the class labels in these data sets as one hidden concept of the data. In addition, we created multiple concepts per object by randomly concatenating objects of different classes, resulting in one high dimensional data set.

To ensure comparability of evaluations, we measure runtimes on identical machines with 2.33GHz Intel XEON CPU, 2 GB of main memory and JAVA 1.6 runtime environment. Furthermore, for comparable quality measurements we use the $F1$ value as in the previous chapter.

4.5.1 Scalability

Database size. In Figure 4.4(a) we analyze the quality of the clustering results with respect to the database size. While increasing the number of objects, we keep the number of concepts fixed to three. We generate concepts with five relevant attributes such that overall we obtain a 15-dimensional data space. Our OSCLU algorithm yields the highest quality compared to all other algorithms, as we detect all hidden clusters in various concepts. The quality of OSCLU is independent of the database size and very robust. StatPC and P3C show good quality results, but also high fluctuating values which cannot reach the quality of OSCLU. All three orthogonal clustering approaches show only low and decreasing quality with respect to the database size. Their underlying k-means model tries to partition the data in each iteration of orthogonal cluster detection. Thus, it cannot cope with the fixed noise ratio in the data which is always assigned to some of the detected clusters and hence resulting in low quality clusterings.



(a) Quality of clustering (b) Runtime

Figure 4.4: Scalability w.r.t. database size

The runtime with respect to the database size is presented in Fig. 4.4(b). The slopes of all curves are in the same range and the influence of the size on all algorithms becomes apparent. The two top quality approaches, our OSCLU model and StatPC, result in similar runtimes. Our redundancy checks and also our density-based model are very complex, but these aspects account for the high quality. The remaining algorithms are faster but we believe, that our runtime is still acceptable considering our high quality results. Especially with increasing concept number, as presented in the next experiment, our model outperforms all other approaches.

Number of concepts. The aim of our model is the detection of multiple concepts, which arise in real scenarios. Thus, in the next experiment we analyze the performance of the algorithm by increasing the number of concepts hidden in a database. To scale the number of concepts, we use a simple data set with only 1000 objects as most of the algorithms showed comparable quality values in this range in the previous experiment. We vary the number of hidden concepts in Figure 4.5 from 1 to 5.

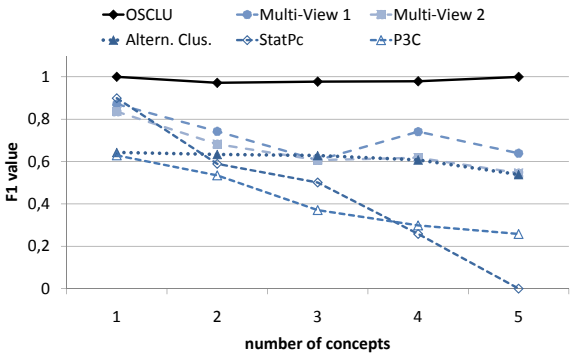


Figure 4.5: F1 vs. number of concepts

We show that OSCLU is able to detect clusters even if objects cluster in multiple concepts. It shows high quality even for a high number of hidden concepts. While traditional clustering approaches aim at clustering single concept data, the alternative clustering approach is designed for two concepts and the multi-view approaches should detect even more than two. However, even these approaches cannot compete with our model. For the subspace and projected clustering approaches, increasing number of concepts makes it very hard to detect the hidden clusters. Especially the projected clustering approach P3C shows decreasing quality, as each object belongs to at most one concept. Overall, StatPC and P3C are not able to detect the multiple hidden concepts per object, while OSCLU yields very high clustering quality.

Noise percentage. In the previous experiments we showed that we outperform subspace and projected clustering approaches as they cannot cope with multiple hidden concepts. Thus, in the following experiments we focus on a more detailed comparison of OSCLU against the orthogonal clustering techniques detecting multi-view and alternative clusterings. First we analyze the effect of noisy data especially for high concept numbers. For the next experiment, illustrated in Figure 4.6, we

generate data with five hidden concepts and vary the noise percentage. On such a difficult data setting, our OSCLU approach outperforms the other techniques. It can detect the clusters hidden in different concepts even in very noisy data sets. Both multi-view algorithms and the alternative clustering approach show again decreasing qualities.

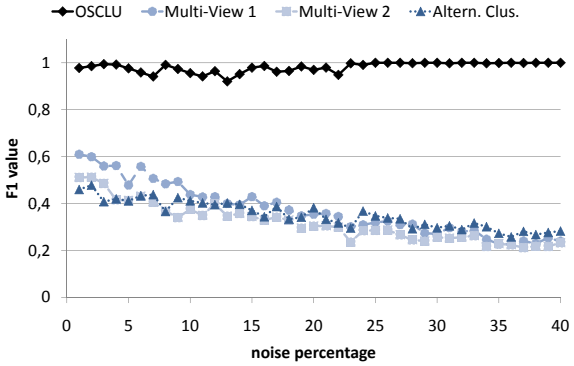
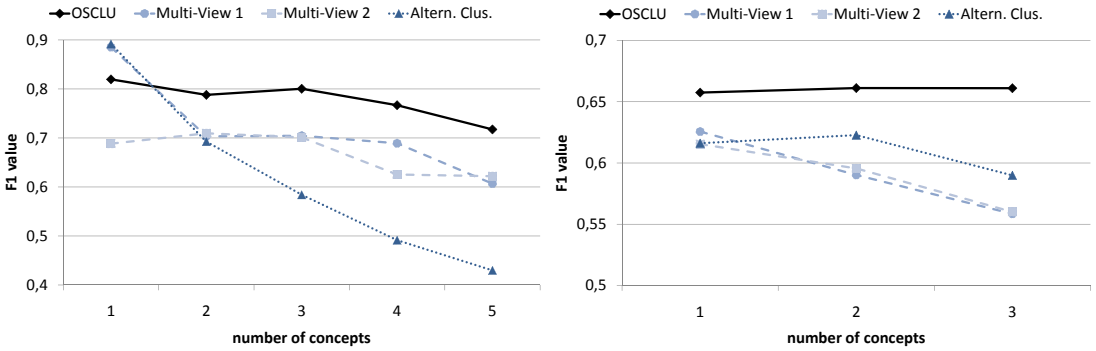


Figure 4.6: F1 vs. noise

4.5.2 Real world data

As we aim at detection of multiple concepts we focus our evaluation for real world data also on scalability w.r.t. number of concepts. We use single concept data obtained by UCI repository and extend them to multi concept data sets. As described in the experiment set-up, similar to synthetic data we can vary the complexity of data sets by including more and more hidden concepts. However, in contrast to the previous experiments we use real world data distribution for the single concepts. We evaluate the effect of variable concept counts on the clustering quality, as for an increasing number of concepts it is more difficult for all algorithms to identify the hidden structure of the data.



(a) Data set: multi-concept iris (b) Data set: multi-concept liver disorder

Figure 4.7: Quality on extended real world data sets with increasing number of concepts

In Figure 4.7 we show clustering quality on iris and liver disorder data set. For the very simple case of only one concept (original UCI data sets), the quality is

high for all algorithms. However, for increasing number of hidden concepts, quality dramatically drops for all competing approaches. Especially, quality of the alternative clustering approach drops with more than two concepts, as it is designed for up to two concepts only. OSCLU shows significantly better performance as it keeps quality high, outperforming the competing approaches for multiple concept data. Although we set for higher number of concepts the optimal parameter value k such that the number of found clusters corresponds to the number of hidden clusters, the competing approaches are not able to detect all hidden concepts. Thus, our OSCLU approach clearly outperforms all competing algorithms even for increasing number of concepts per object.

4.5.3 Parameterization

In addition to the experiments that compare OSCLU to existing methods, we analyze the flexibility of our model. As presented in Sec. 4.3.2 the user can control the output by changing the required interestingness notion. As main parameters α and β control the allowed overlap in terms of objects (α) and in terms of dimensions (β). The intended effects are comparable for object and dimension overlap respectively, thus we show the resulting effects only for parameter α. In Figure 4.8, we present the variation of the parameter α which controls the interestingness of each cluster in the result set. As intended by this parameter, higher values of α increase the required interestingness (each cluster has to provide more novel objects to become interesting and appear in the output) and hence less clusters are in the result. By varying the α parameter one can control the overall result set based on our global interestingness I_{global}. We include a cluster only if the fraction of its newly clustered objects is at least α (cf. Def. 17). Furthermore, our OSCLU algorithm is not only able to detect orthogonal concepts, but in addition it is very flexible by using a local interestingness I_{local}. It allows the user to control the output dependent on the application or the current interestingness.

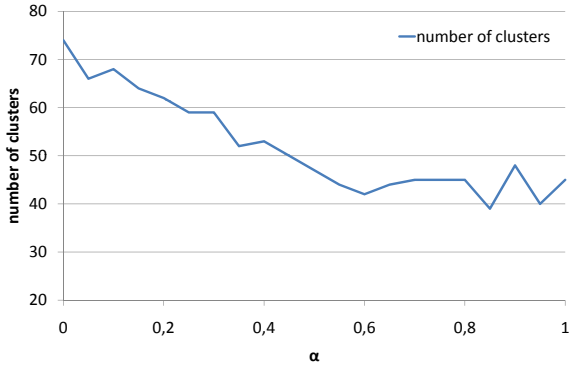


Figure 4.8: Variation of global interestingness

4.6 Enhancements by detection of orthogonal subspace clusters

We introduced the OSCLU (Orthogonal Subspace CLUstering) approach. It overcomes major drawbacks of existing approaches in the detection of multiple concepts hidden in arbitrary subspace projections of the data. First, our novel clustering model actively searches for multiple concepts per object (cf. Challenge 2). Second, based on a variable density measure in the instantiation of our subspace cluster definition our model adapts to arbitrary subspaces (cf. Challenge 1). And third, it computes an optimal orthogonal clustering by ensuring non-redundancy and maximal interestingness of the resulting clustering (cf. Challenge 3).

Furthermore, we show that our clustering model is NP-hard and propose an efficient approximative algorithm. We approximate the optimization problem by pruning similar subspaces ensuring efficient cluster detection in only the orthogonal subspaces. Thus, our OSCLU approach is the first method for detection of multiple orthogonal concepts in subspaces of high dimensional data. It provides novel information about the hidden structure of the data as each detected concept is described by orthogonal subspaces while OSCLU prunes redundant concepts in similar subspaces. Thorough experiments demonstrate that OSCLU clearly outperforms existing subspace clustering and orthogonal clustering algorithms while automatically reducing the output to only the orthogonal concepts hidden in the data.

Overall, the OSCLU model copes with all challenges for high quality subspace clustering mentioned in Section 1.3. It shows high potential to be the fundamental subspace clustering model in future research. Based on the mentioned subspace clustering models in this thesis, there are some further open challenges derived by our work that will be mentioned in Section 15.3. However, we close the modeling part of this thesis as further algorithmic enhancements are clearly required for all of the proposed subspace clustering models. It is important to discuss such algorithmic developments, as efficient computation of complex but high quality clustering models is crucial for their scalability to real world applications. In the following part of this thesis we will go into details for some of our general techniques providing efficiency improvements in subspace clustering.

Part II

Efficient subspace cluster computation

Chapter 5

Efficient multi-step architecture for subspace clustering

In the first part of this thesis we proposed several subspace clustering models that have shown to mine high quality clusters hidden in subspaces of high dimensional databases. The main focus was on the quality of subspace clustering results. For each of these approaches we have also considered efficient algorithms to compute these high quality subspace clusters. For example, we have developed relaxations of our optimization models for an efficient approximation of these NP-hard problems. In the second part we propose several general solutions for further efficiency enhancements. We tackle the high cost of density-based subspace clustering by reducing the exponential search space but also the high computational cost of neighborhood computations in the density-based paradigm. We introduce a novel depth-first processing for efficient in-process removal of redundancy and propose a novel index structure for subspace mining. Furthermore, for our optimization models we propose efficient processing schemes that overcome drawbacks of traditional bottom-up processing.

As first technique, we propose EDSC, a lossless filter architecture reducing costly database scans. Our EDSC (efficient density-based subspace clustering) algorithm reduces the high computational cost of density-based subspace clustering by a complete multi-step filter-and-refine algorithm. Our first hypercube filter step avoids exhaustive search of all regions in all subspaces by enclosing potentially density-based clusters in hypercubes. Our second filter step provides additional pruning based on a density monotonicity property. In the final refinement step, the exact unbiased density-based subspace clustering result is detected. As we prove that pruning is lossless in both filter steps, we guarantee completeness of the result. In thorough experiments on synthetic and real world data sets, we demonstrate substantial efficiency gains. Our lossless EDSC approach outperforms existing density-based subspace clustering algorithms by orders of magnitude. Thus, it forms the basis for further development of efficient subspace clustering algorithms for our more complex clustering models as we will show in the following chapters.

5.1 Motivation and comparison with related work

In general, subspace clustering is a computationally challenging task as one searches for clusters hidden in any possible subspace projection. Most approaches propose a trade-off between efficient computation and high quality results. Focusing on efficient subspace clustering, grid-based discretization of the space or other lossy approximations have been proposed [AGGR98, NGC01, SZ04]. While these algorithms show good runtimes, they lose clusters which are cut apart by the grid or missed by the approximation.

As the first subspace clustering approach, CLIQUE uses a grid to discretize the search space [AGGR98]. Grids greatly reduce the computational complexity, yet clusters which spread across cells are missed and results are sensitive to the position of the grid. Extensions of this approach have tried to tackle this challenge by flexible grid positioning [NGC01] or by variable thresholds [SZ04]. However, heuristics and a grid-based discretization are used for pruning. Consequently, clusters are lost as well as in any existing grid-based discretization. In general, apriori-based subspace clustering algorithms prune based on a monotonicity assumption of subspace clusters with respect to the dimensionality [AGGR98, KKK04]. However, the incurred dimensionality bias as described in Chapter 2 leads to loss of high dimensional subspace clusters. Consequently, approximate, grid-based, and biased algorithms all fail to detect all subspace clusters.

Density-based approaches define clusters as dense areas separated by sparsely populated areas and have been shown to successfully mine arbitrarily shaped clusters even in the presence of noise [EKSX96, KKK04]. However, their runtimes are significantly higher due to repeated neighborhood density computations, making them infeasible for many practical applications [KKK04]. Density-based subspace clustering approaches overcome grid-based cluster loss [KKK04]. Runtimes are better than for naive re-runs of DBSCAN, yet still not feasible for practical applications. Moreover, the assumption of density monotonicity still results in losing subspace clusters in higher dimensions. Our unbiased density-based clustering model overcomes this dimensionality bias by adapting density assessment to the dimensionality of the subspace (cf. Chapter 2). Consequently, based on the density-based paradigm it is capable of detecting arbitrarily shaped subspace clusters of any dimensionality. As density monotonicity does not hold for unbiased density, it requires novel techniques for efficient computation.

In this work, we propose a concept for overcoming the existing trade-off between accuracy and efficiency. We present a density-based subspace clustering algorithm EDSC (efficient density-based subspace clustering) in a multi-step filter-and-refine architecture. Substantial efficiency gains are achieved by two novel filter steps. The first step reduces the search space by efficiently mining candidate subspace clusters in enclosing hypercubes. This hypercube filter reduces the number of database scans for density computations and, via our proven hypercube monotonicity, allows for effective and lossless pruning of hypercubes in many irrelevant subspace projections. For the second step, a novel density filter that further prunes subspace cluster candidates without loss of completeness, we prove a monotonicity property for unbiased

density approaches. Both filter steps significantly reduce the number of subspace cluster candidates without loss of results. The final refinement step ensures that the exact density-based subspace clustering result is found, yet with a substantially lower runtime. Our multi-step approach thus efficiently detects density-based subspace clusters even in large high dimensional data spaces.

Summarizing, EDSC shows two core characteristics:

- **Accuracy**: density-based subspace clustering with guaranteed lossless pruning
- **Efficiency**: efficient subspace clustering via a novel multi-step filter-and-refine algorithm

5.2 Efficient multi-step architecture

Detecting all subspace clusters in any subspace is a highly complex task. The number of possible subspaces is exponential in the number of attributes. Naively searching all of these subspaces is hence not feasible in most applications and pruning the search space is crucial. In this section, we propose our efficient and lossless density-based subspace clustering algorithm EDSC (efficient density-based subspace clustering).

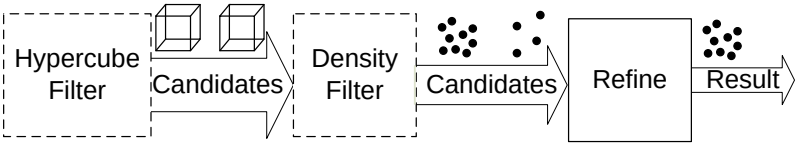


Figure 5.1: Multi-step EDSC algorithm

5.2.1 The multi-step architecture

We propose a multi-step filter-and-refine architecture as illustrated in Figure 5.1 for efficient subspace clustering. Instead of running a complex subspace clustering algorithm on the entire database, we restrict the search space in a chain of filter steps. Each filter step reduces the search space to a successively smaller set of candidates, i.e. sets of objects that are potential density-based subspace clusters. EDSC uses two novel filters for both complete and selective pruning to ensure that the overall algorithm is accurate and efficient. Completeness means that the filters do not produce any false dismissals, i.e. we guarantee that all subspace clusters are included in the candidate sets, and that the result set of the EDSC algorithm contains all subspace clusters. Selective pruning means that the algorithm achieves a large reduction in the search space, i.e. many irrelevant sets of objects and their higher dimensional projections can be excluded from further consideration for substantial speed-up. The combination of both filters in EDSC is illustrated in Figure 5.2: from the fact that a candidate fails one of the two filter tests, we infer that it is not a subspace cluster in any higher dimensional subspace.

The refinement step additionally ensures that there are also no false positives, i.e. only those candidates are actually reported as results that constitute subspace clusters.

Hypercube filter. Existing density-based subspace clustering algorithms suffer from the fact that for each subspace and for each object, density has to be computed. This requires repeated neighborhood scans of the database. Our first filter step thus determines hypercubes that surround each potential subspace cluster, thus reducing the neighborhood evaluation to significantly fewer objects. The algorithm then processes only these candidate in higher dimensional projections and thus can efficiently reduce the exponential search space.

Density filter. In the second filter, we build on the hypercube restriction from the first step. The basic idea is to determine for any candidate hypercube not only if it potentially contains a density-based subspace cluster but also if any of its higher dimensional projections may contain one. For unbiased density (cf. Def. 5), monotonicity does not hold and thus cannot be used for lossless pruning (see Section 5.2.3 for details). However, for our density filter we based on a derived monotonicity property that allows for lossless pruning even with respect to this unbiased density criterion. This *weak density*, which determines a "minimum" density over all subspaces, ensures that we may safely prune a candidate hypercube and all its higher dimensional projections, without loss of completeness. Our *weak density* filtering precedes the final refinement step.

Refinement. Subspace clusters which fulfill the "minimum" density criterion (pass the *weak density* filter) are now iteratively processed in higher dimensional subspaces to finally determine the exact result according to the unbiased variable density definition. The refinement step thus removes any remaining false alarms as it uses the stricter unbiased density.

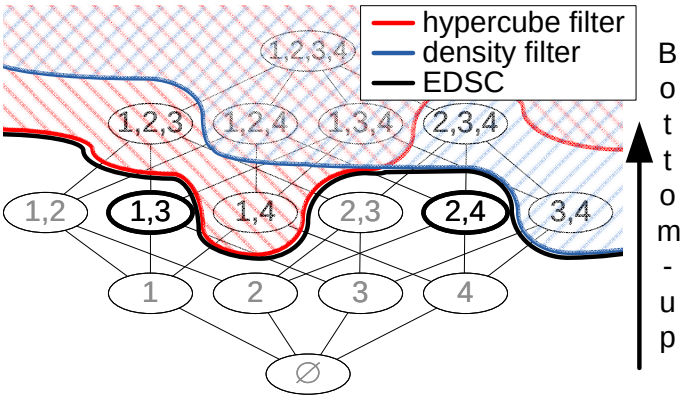


Figure 5.2: Pruning the search space

5.2.2 Hypercube filter

To avoid repeated scans of the entire database to determine the neighborhood of objects, we enclose potential subspace cluster regions in hypercubes.

Building hypercubes With the novel density-conserving grid we guarantee to enclose cluster regions and thus achieve a lossless and efficient pruning. The density-conserving grid is devised to meet exactly the definition of density-based subspace clusters. Defined as sets of density-connected objects w.r.t. their ε-neighborhoods, subspace clusters might spread across multiple grid cells. To detect these clusters, we introduce novel connectivity borders of ε width. Subspace clusters which are density-connected across cells can be detected along the connectivity border between cells. In the algorithm, these cells are merged until completely enclosing hypercubes for each subspace cluster are found.

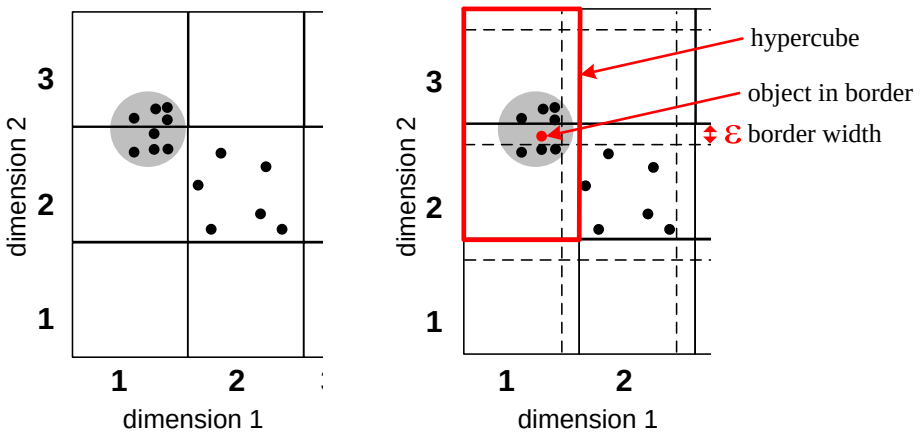


Figure 5.3: Traditional vs. density-conserving grid

Figure 5.3 illustrates the idea: on the left side, we show a traditional grid structure. Traditional grids loose clusters that spread across several cells. For example, by simply comparing the count of objects in each cell with an exemplary threshold of 5 objects, the dark shaded cluster at the left hand side would be missed, and only the objects in cell $(2, 2)$ would be reported as a cluster. Our novel density-conserving grid, illustrated at the right side, has additional borders of the size of the neighborhood, ε. Checking these borders, our EDSC algorithm detects all potential spreads of a cluster from one cell to another and merges the corresponding cells to build an enclosing hypercube. Thus, the dark shaded cluster is successfully detected.

Formally, our density-conserving grid is a partitioning of the data space into regular grid cells, plus novel connectivity borders that are exactly the size of ε-neighborhoods that are used in the definition of density-connected objects (Def. 5). Each cell is identified by the indices of the cell intervals in each dimension. Borders are additionally marked for later processing in our algorithm:

Definition 19. Density-conserving grid.

A **density-conserving grid** *is a regular grid with connectivity borders:*

- *A* **regular grid** *is a partition of the attribute range v into $g = \lceil \frac{v}{w} \rceil$ intervals p_i of equal width w and*
  $p_i = [w \cdot (i-1), w \cdot i),\ i = 1 \ldots g.$
- **Connectivity-borders** *are intervals of ε-width at the upper border of each cell in each dimension*
  $b_i = [w \cdot i - \varepsilon, w \cdot i),\ i = 1, \ldots, g$
- *A s-dimensional* **subspace cell** $C_{\alpha_1,\ldots,\alpha_d}$ *is given by an index vector $\alpha_1, \ldots, \alpha_d$ of the corresponding intervals p_{α_j}, $\alpha_j \in \{1, \ldots, g, *\}$, where stars denote the unconstrained dimensions of the cell.*
- *A border $B_{\alpha_1,\ldots,\overline{\alpha_k},\ldots,\alpha_d}$ in dimension k is given by the index vector $\alpha_1, \ldots, \overline{\alpha_k}, \ldots, \alpha_d$ of its cell $C_{\alpha_1,\ldots,\alpha_d}$, where the dash denotes the border dimension k.*

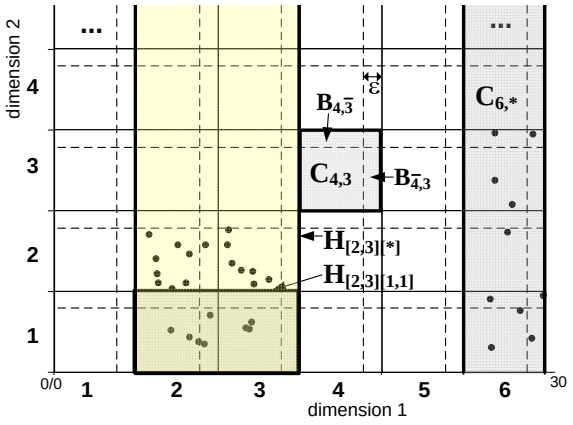


Figure 5.4: Density-conserving grid and hypercubes

In Figure 5.4 we illustrate our approach by an example of a two-dimensional space where each attribute range v is 0 to 30, and the grid resolution g is set to 6. For example, cell $C_{4,3}$ contains two connectivity borders $B_{\overline{4},3}$ and $B_{4,\overline{3}}$, one in each dimension. In the example, $C_{4,3}$ is a 2-dimensional cell which is restricted in interval 4 for dimension 1 and in interval 3 for dimension 2. $C_{6,*}$ is a 1-dimensional cell restricted in interval 6 for dimension 1 and not constrained in dimension 2. In general $(d - |S|)$ stars denotes the unconstrained dimensions of the cell. Hypercubes consist of merged cells, given by interval ranges per dimension. $H_{[2,3][1,1]}$ is a 2-dimensional example hypercube. $|H_{[a_1,b_1]\ldots[a_d,b_d]}|$ denotes the number of objects in a hypercube. For example, $H_{[2,3][*]}$ contains 24 objects (all objects contained in the 2nd and 3rd interval), while its constrained projection to interval one in dimension one $|H_{[2,3][1,1]}|$ contains 8 objects.

This density-conserving grid is now used to efficiently construct hypercubes that enclose potential density-based subspace clusters. Thus, we avoid repeated clustering on individual objects in each subspace.

Filtering using hypercubes The hypercube filter on this density-conserving grid works as follows: all grid cells are accessed exactly once, by processing them in lexicographic order on their interval indices (e.g. in Figure 5.4, $C_{1,1}, C_{1,2}, \ldots, C_{6,5}, C_{6,6}$). For each non-empty cell, its borders are checked to see whether a subspace cluster might spread across the cell borders. As the size of the border is exactly the size of the neighborhood in the density-connectivity definition, any such spread is guaranteed to be detected by checking all borders of the cell. Only objects in the borders could connect a cluster from one cell to the next, otherwise the density-connectivity property is not fulfilled. If the borders are empty, the hypercube is maximal and encloses the entire potential subspace cluster. We denote it as MICH (*maximal induced cluster hypercube*) and continue with the next filter step. If the borders of the cell are not empty, we repeatedly merge hypercubes (details in Section 5.2.4) and check borders until the MICH is maximal. As our hypercube filter checks all cells and all connectivity borders between them, EDSC is guaranteed to find all MICHs, i.e. all potential subspace clusters.

Theorem 3. Completeness of hypercube filter

Each subspace cluster C is enclosed by a MICH.

Proof.
By Definition 5, C is a set of maximal density-connected objects $Q_1, \ldots, Q_n$.
Case 1: All density-connected objects are within one cell. Thus an index vector $\alpha_1, \ldots, \alpha_d$ exists such that all objects are in the corresponding grid cell $\forall i \in \{1, \ldots, n\} : Q_i \in C_{\alpha_1,\ldots,\alpha_d}$. The cell $C_{\alpha_1,\ldots,\alpha_d}$ forms a MICH and encloses all objects in C.
Case 2: All density-connected objects are within two neighboring cells. Thus there are two directly density-connected objects Q_i and Q_{i+1} positioned in two neighboring grid cells. Furthermore, at least one dimension l exists, in which these cells differ, w.l.o.g. $\beta_l = \alpha_l + 1$. In our grid cell notation, we have:

Q_i	$\in$	$C_{\alpha_1,\ldots,\alpha_d}$	left cell
Q_{i+1}	$\in$	$C_{\beta_1,\ldots,\beta_d}$	right cell
p_{α_l}	$=$	$[w \cdot (\alpha_l - 1), w \cdot \alpha_l)$	left interval
b_{α_l}	$=$	$[w \cdot \alpha_l - \varepsilon, w \cdot \alpha_l)$	border interval
p_{β_l}	$=$	$[w \cdot \alpha_l, w \cdot (\alpha_l + 1))$	right interval

The necessary condition for merging in EDSC is that there is an object in the connectivity-border.
Therefore, we proof $Q_i \in B_{\alpha_1,\ldots,\overline{\alpha_l},\ldots,\alpha_d}$ by contradiction: Assume $Q_i \notin B_{\alpha_1,\ldots,\overline{\alpha_l},\ldots,\alpha_d}$ and hence the distance of Q_i to the upper limit of the grid cell intervall p_{α_l} is greater than ε. Hence the overall distance of Q_i to any object in the interval p_{α_l+1} is also larger than ε. Formally, $dist(Q_i, Q_{i+1}) \geq dist(Q_i^{\{l\}}, w \cdot \alpha_l) > \varepsilon \Rightarrow Q_i \notin N_\varepsilon(Q_{i+1})$. This is a contradiction to the density-connectedness of Q_i and Q_{i+1}.

As Q_i is in the connectivity-border $B_{\alpha_1,\ldots,\overline{\alpha_l},\ldots,\alpha_d}$ of cell $C_{\alpha_1,\ldots,\alpha_d}$, the merge is detected when processing $C_{\alpha_1,\ldots,\alpha_d}$. EDSC will therefore merge the cells $C_{\alpha_1,\ldots,\alpha_d}$ and $C_{\beta_1,\ldots,\beta_d}$ to an enclosing MICH.

Case 3: For density-connected objects within more than two neighboring cells, the argument of Case 2 for directly connected neighboring Q_i, Q_{i+1} can be inductively extended to all objects in the density-connected set. Therefore, the entire density-connected subspace cluster $Q_1, \ldots, Q_n$ is detected through border checks, and the enclosing MICH M_C is built. □

For good runtime performance it is important to prune hypercubes that cannot contain subspace clusters. We propose pruning those hypercubes that do not fulfill minimum size requirements for density-based subspace clusters. As we search for clusters with at least $minSize$ objects, monotonicity with respect to the number of objects allows for safe pruning. We exploit monotonicity on the number of objects in hypercubes of different subspaces. Monotonicity means that as we restrict a hypercube in more dimensions the number of objects cannot increase. Therefore, a hypercube which does not contain enough objects in a subspace S does not contain enough objects in any higher dimensional subspace T ($S \subseteq T \subseteq DIM$), either. Thus, we may safely prune this hypercube from further consideration without losing any subspace cluster.

Theorem 4. Hypercube monotonicity.
For any two subspaces with $S \subseteq T$, and any hypercube H identified by interval indices $[a_1, b_1] \ldots [a_d, b_d]$ the number of objects in T is bound by the number of objects in S:

$$|H^S_{[a_1,b_1]\ldots[a_d,b_d]}| \geq |H^T_{[a_1,b_1]\ldots[a_d,b_d]}|$$

Proof.

$$\begin{aligned} &|H^S_{[a_1,b_1]\ldots[a_d,b_d]}| \\ &\quad = |\{(o_1, \ldots, o_d) \in DB \mid \forall i \in S : o_i \in [w \cdot a_i, w \cdot b_i)\}| \\ &\quad \geq |\{(o_1, \ldots, o_d) \in DB \mid \forall i \in T : o_i \in [w \cdot a_i, w \cdot b_i)\}| \\ &\quad = |H^T_{[a_1,b_1]\ldots[a_d,b_d]}| \end{aligned}$$

As $S \subseteq T$ holds, their are more constraints on objects in subspace T. And therefore in the hypercube $H^T_{[a_1,b_1]\ldots[a_d,b_d]}$ there are at most as many objects as in $H^S_{[a_1,b_1]\ldots[a_d,b_d]}$. □

As we have proven in Theorem 3, each density-based subspace cluster is enclosed by a MICH. Thus our EDSC algorithm may safely prune sparse hypercubes and all its higher dimensional projections. Any MICH $H_{[a_1,b_1]\ldots[a_d,b_d]}$ is checked in subspace S to see if $|H^S_{[a_1,b_1]\ldots[a_d,b_d]}| \geq minSize$. If this is the case, we continue with this candidate. Else, we have detected a spare region and we know for sure that we can not only discard $H^S_{[a_1,b_1]\ldots[a_d,b_d]}$, but also all higher dimensional projections $H^T_{[a_1,b_1]\ldots[a_d,b_d]}$ with $S \subseteq T$, since for them $|H^T_{[a_1,b_1]\ldots[a_d,b_d]}| \geq minSize$ cannot hold either.

5.2.3 Density filter

MICHs are conservative approximations of potential subspace clusters, i.e. all subspace clusters are enclosed by one MICH, but not all MICHs contain subspace clusters. To efficiently detect those sets of objects that are not dense and thus do not form

subspace clusters, we devise our density filter. It prunes those subspace clusters that do not fulfill minimum density in the current and also all higher dimensional subspaces. As we prove completeness also for our second filter, overall lossless pruning is ensured.

Density monotonicity In fixed threshold subspace clustering (more objects in the neighborhood than $minPoints$), density is monotonous with respect to the number of dimensions which is used for pruning as in SUBCLU [KKK04]. Those sets of objects that are not dense in a certain subspace, may not be dense in any higher dimensional subspace, either, and are consequently pruned. In principle, we could use the same pruning idea. However, as shown in our previous work, this fixed threshold leads to dimensionality bias, as the expected density decreases rapidly with increasing dimensionality. Thus, high dimensional subspace clusters are almost never detected. We therefore use unbiased density as used in Definition 5 (density larger than F times the expected density). While this definition allows for detection of clusters in arbitrary subspaces, it is no longer monotonous, as the expected density decreases with growing dimensionality. For completeness, we therefore propose using a weaker criterion for pruning based on monotonicity of the number of objects in the neighborhood with increasing dimensionality.

Theorem 5. Density monotonicity.

For any two subspaces with $S \subseteq T$, and any object O the number of objects in the neighborhood in subspace T is bound by the number of objects in subspace S:

$$|N_\varepsilon^S(O)| \geq |N_\varepsilon^T(O)|$$

Proof.

The proof is straightforward from the definition of the neighborhood. It uses the fact that with more dimensions in T, the distances of objects are larger than in S:

$$\begin{aligned} dist^S(O,P) &= \sqrt{\textstyle\sum_{i\in S}(o_i - p_i)^2} \\ &\leq \sqrt{\textstyle\sum_{i\in T}(o_i - p_i)^2} = dist^T(O,P) \\ |N_\varepsilon^S(O)| &= |\{P \in DB \mid dist^S(O,P) \leq \varepsilon\}| \geq \\ &\quad |\{P \in DB \mid dist^T(O,P) \leq \varepsilon\}| = |N_\varepsilon^T(O)| \end{aligned}$$

□

Pruning with this criterion for unbiased density measures (cf. Def. 5) would violate completeness. The expected density decreases with increasing dimensionality. If we were to discard a set of objects that does not exceed the $F \cdot expDen(s)$ threshold for a dimensionality s, it can still exceed the lower threshold $F \cdot expDen(s')$ for $s' \geq s$. Pruning based on s alone would therefore lose clusters. To allow for lossless pruning even for algorithms with unbiased density measurements, our "weak density" criterion is derived out of the unbiased density measure as described in Section 2.4. It is based on the lowest possible expected density in the highest dimensionality d.

Definition 20. Weak Density.

An object $O \in DB$ is weak dense in S if:

$$|N_\varepsilon^S(O)| \geq max\{minPoints, F \cdot expDen(d)\}$$

Thus, density in the subspace with the lowest expected density leads to a complete filter:

Theorem 6. Completeness of density filter
For all subspaces T with $S \subseteq T \subseteq DIM$ and $|T| = t$ holds:

If an object is not **weak dense** *in S*
it is not **dense** *in T.*

Proof.

$$|N_\varepsilon^S(O)| < max\{minPoints, F \cdot expDen(d)\} \quad (5.1)$$
$$\Rightarrow |N_\varepsilon^T(O)| < max\{minPoints, F \cdot expDen(d)\} \quad (5.2)$$
$$\Rightarrow |N_\varepsilon^T(O)| < max\{minPoints, F \cdot expDen(t)\} \quad (5.3)$$

Using monotonicity property (Theorem 5) the left side of inequality ($1 \rightarrow 2$) is monotonically decreasing with increasing dimensionality. As the expected density ($expDen$) is calculated as the ratio of the volume of the ε-sphere to the volume of the subspace, $expDen$ is monotonously decreasing with increasing dimensionality. Hence, the subspace of the highest dimensionality d has the lowest density: $expDen(d) \leq expDen(t)$. Thus the maximum on the right side of inequality (2) is less than or equal to the maximum in inequality (3). □

Following the weak density theorem, an object can be pruned if it violates the weak density condition, as it is not dense in any higher dimensional subspace. Hence we check any candidate that passed the hypercube filter according to the weak density criterion in the density filter. If it is no subspace cluster with respect to both $minPoints$ and $F \cdot expDen(d)$, we may safely discard all its objects, as we know for sure that they do not constitute a subspace cluster with respect to unbiased density $F \cdot expDen(t)$ and $minPoints$ in any dimensionality t.

This concludes the discussion of the two filters in our multi-step EDSC approach and the completeness proof. In the next section, we discuss the overall algorithm as illustrated in Figure 5.1 and provide more detailed information on efficient handling and merging of hypercubes.

5.2.4 The EDSC algorithm

Algorithm 3 outlines the complete algorithm that consist of hypercube filter, density filter and refinement (Fig. 5.1).

Algorithm overview Recall that each MICH encloses a potential density-based cluster (a subspace cluster candidate). Our algorithm processes each such MICH in all subspace projections. Thus, the **hypercube filter** (Line 1-12) recursively calls the other filter steps of our multi-step algorithm for each MICH. Using the hypercube monotonicity property in Theorem 4 a MICH and all its higher dimensional subspace projections can be pruned if the MICH does not contain at least $minSize$ objects

(Line 14). If a MICH is a cluster candidate, it is subsequently analyzed by the **density filter** of the EDSC algorithm (Line 15) based on the weak density in Theorem 5. If the hypercube does not contain any weak density connected subspace cluster the corresponding hypercube and all its higher dimensional hypercube projections can be pruned. Additionally, redundant clusters are discarded according to the last part of the subspace cluster definition in Def. 5 (Line 17). The **refinement step** removes any false alarms: the method *ExpDensityScan* performs the actual subspace clustering w.r.t. the expected density of the subspaces.

Algorithm 3 EDSC algorithm: EDSC(C, d_{first})

```
1  for k = d_first ... d do                                   /* for each dimension and */
2      for i = 1 ... g do                                     /* each cell in each dimension */
3          C = restrictCell(C, [k, i]);                       /* restrict C in dimension k */
4          if restrictions(C) < 2 then                        /* no testing for one-dimensional cells */
5              EDSC(C, k + 1) ;                               /* continue EDSC recursion */
6          else
7              if restrictions(C) == 2 then
8                  I = testBorder(C) ;                        /* test all borders */
9              else if restrictions(C) > 2 then
10                 I = testNewBorder([k, i]);                 /* test only new border */
11             induceMerge(C, I) ;                            /* for future neighbors of C */
12             C = performMerges(C) ;                         /* for past neighbors of C */
13             if induceCounter(C)==0 then                    /* MICH complete */
14                 if objectCounter(C) > minSize then         /* 1st filter step */
15                     if WeakDensityScan(C) then             /* 2nd filter step */
16                         EDSC(C,k+1);                       /* further restriction */
17                         if isNonRedundant(C) then
18                             ExpDensityScan(C)              /* refinement step */
19                         else
20                             pruneCluster(C) ;              /* prune C */
21                     else
22                         prune(C) ;   /* prune C and all of its higher dim. projections */
23                 else
24                     prune(C) ;       /* prune C and all of its higher dim. projections */
```

Efficient hypercube computation In this section, we give algorithmic details and an illustrative example on how to efficiently create enclosing hypercubes by merging of grid cells.

The algorithm starts on cells that are merged until an enclosing hypercube for each subspace cluster is found. For efficiency reasons we process cells in lexicographical order to ensure that each cell has to be processed only once. We mark future merges w.r.t lexicographic order as *induced merges*; when these marked cells are actually processed, *performed merges* combine the cells. Lexicographically greater cells are processed in the future, hence we denote them as *future cells*, and lexicographically smaller cells as *past cells*. An enclosing hypercube is found if no more induced merges and performed merges are necessary from the current cell.

If a density-connected subspace cluster stretches from one cell into another, the connectivity border contains at least one object. Otherwise, as we set the border width exactly to the neighborhood range ε, objects in one cell cannot be in the ε-neighborhood of the other. When a cell is processed, the connectivity borders of the cell are checked. For each border which contains an object a merge is induced into the corresponding adjacent cell. If both connectivity borders contain an object a cluster might also extend into the diagonal cell, i.e. adjacent to the intersection of both borders, ensuring an induced merge to this diagonal cell.

When processing a cell, first merges into future cells are induced and then merges with past cells are performed. A merge is always performed with a past cell which induced a merge into the cell. During a merge process the number of objects (the *objectCounter*) and the number of merges induced into future cells (the *induceCounter*) are aggregated. We use the concept of induced merges to indicate whether an enclosing hypercube for a density-connected region is found: if all induced merges of a hypercube are performed ($induceCounter = 0$) a MICH is found.

Checking connectivity borders guarantees that the grid does not cut density-based clusters apart and that each subspace cluster is enclosed by a MICH (see Theorem 3).

Merge example Figure 5.5 illustrates the merge steps performed to identify a MICH enclosing the example cluster. We start by processing each cell of dimensionality two, beginning with cell $C_{1,1}$. The number of objects is determined, and if the cell is not empty, its connectivity borders are tested. Cell $C_{1,1}$ is empty and hence no merge is induced. Next, cell $C_{2,1}$ is processed. As both connectivity borders contain an object, cell $C_{2,1}$ induces a merge into $C_{3,1}$, $C_{2,2}$ and into $C_{3,2}$. Next, cell $C_{3,1}$ performs the merge with $C_{2,1}$, creating $H_{[2,3][1,1]}$. When $C_{2,2}$ is processed, it is merged with $C_{1,2}$ to $H_{[1,2][2,2]}$. After this, the next merge induced into $C_{2,2}$ is performed (second part of Figure 5.5). This merge step creates the hypercube $H_{[1,3][1,2]}$. Finally, cell $C_{3,2}$ is processed. As $C_{3,2}$ was already merged with both cells, only the *induceCounter* is decremented. After this step, no more merges are necessary and $H_{[1,3][1,2]}$ corresponds to a MICH.

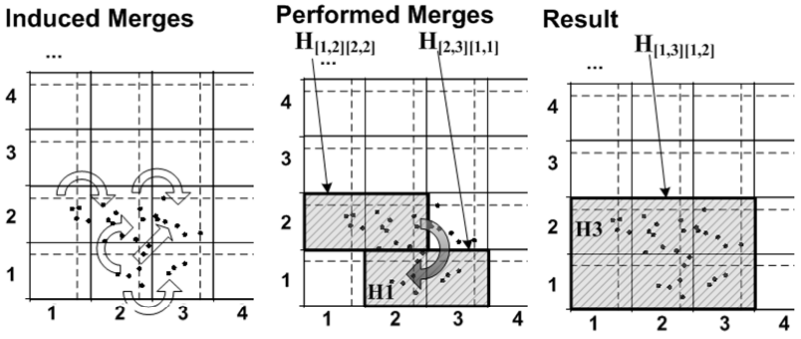


Figure 5.5: Induced and performed merges

After a MICH is found the EDSC algorithm checks if the hypercube contains more

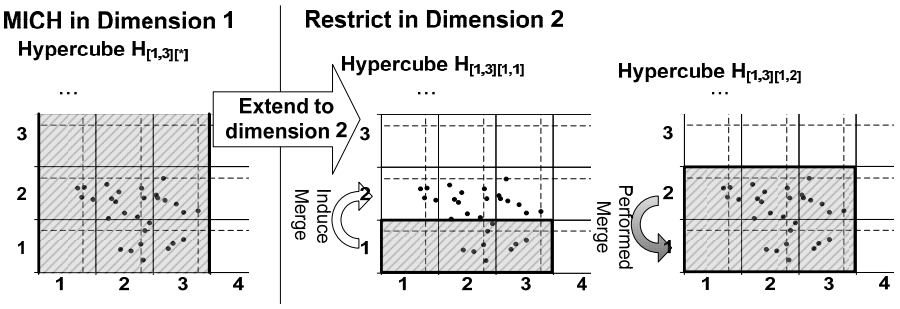


Figure 5.6: Extending a MICH to next dimension

than $minSize$ objects. If the region does not contain enough objects, the hypercube and all higher dimensional projections of that hypercube are pruned (monotonicity property, Theorem 4). Thus this **hypercube filter** step based on merges in the density-conserving grid reduces the number of time consuming database scans for density-connected clusters and to prune the search space.

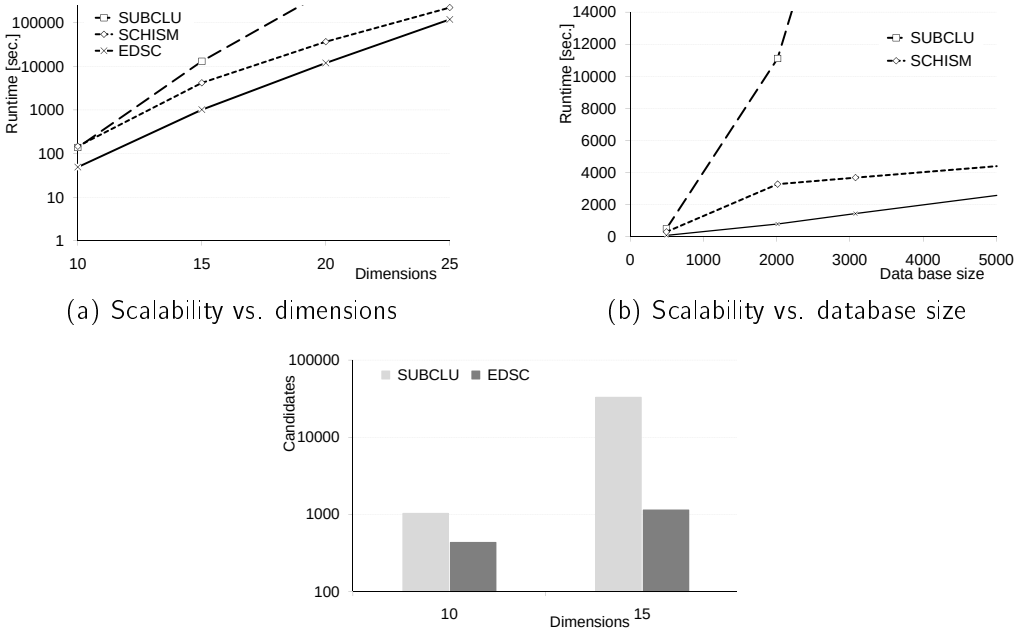
Additionally our EDSC algorithm analyzes only one MICH at a time by successively extending the MICH in all dimensions. When a MICH is extended to a new dimension it is iteratively restricted to each grid cell. For example a two dimensional hypercube embedded in a four dimensional space $H_{[1,3][1,2][*][*]}$ can be extended into dimensions three and four. We first extended it in dimension three and restrict it to cell one ($H_{[1,3][1,2][1,1][*]}$). Afterwards the merge procedure is applied recursively. Assume the merge step mines the MICH $H_{[1,3][1,2][1,2][*]}$. This MICH is next restricted in dimension four: $H_{[1,3][1,2][1,2][1,1]}$. After this step the next hypercube ($H_{[1,3][1,2][*][1,1]}$) is analyzed and processed accordingly.

For simplicity, we demonstrate the extension step using a one-dimensional MICH $H_{[1,3][*]}$ (see Figure 5.6). After extending the hypercube $H_{[1,3][*]}$ to dimension two and restricting it to the first grid cell $H_{[1,3][1,1]}$ the merge procedure is applied again. As the corresponding connectivity border is not empty, a merge is induced and performed directly afterwards, resulting in hypercube $H_{[1,3][1,2]}$. $H_{[1,3][1,2]}$ again corresponds to a MICH in subspace $S = \{1, 2\}$ as all of its connectivity borders are empty.

This concludes the discussion of our EDSC algorithm. In the following, we demonstrate the efficiency and accuracy of our approach in the experimental evaluation.

5.3 Experiments

We evaluate the efficiency of the EDSC algorithm as the first lossless but also efficient approach to density-based subspace clustering. The most recent non-approximate density-based subspace clustering algorithm SUBCLU, which extends DBSCAN to subspace clustering, is used for comparison [KKK04]. Additionally, we contrast our approach with SCHISM, a recent efficient but approximative grid-based algorithm. Experiments were run on Pentium 4 machines with 2.4 Ghz and 1 GB main memory. To evaluate scalability we use synthetic data sets. Further on we show the performance of EDSC on three real world data sets.



(a) Scalability vs. dimensions

(b) Scalability vs. database size

(c) Selectivity vs. dimension

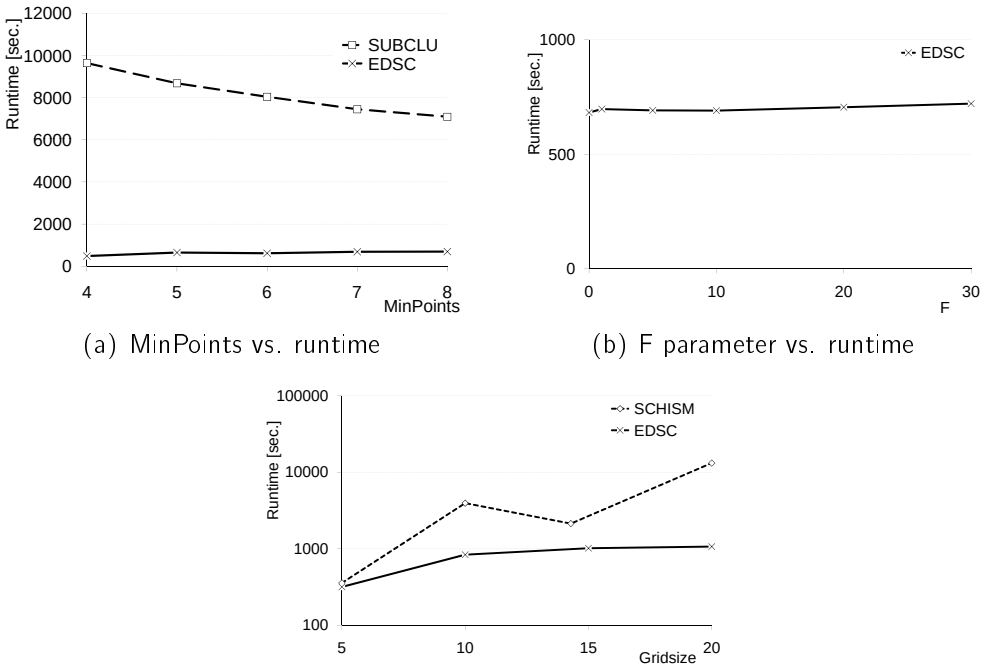
Figure 5.7: Scalability evaluation. Parameters: $gridSize = 10; minSize = 5\% \cdot |DB|; F = 1; minPoints = 8; \varepsilon = 4$

5.3.1 Scalability

Based on properties of real world data sets we generate synthetic data for scalability experiments as in the previous chapters.

Scalability w.r.t. dimensionality. As the number of possible subspace clusters depends exponentially on the dimensionality of the subspace, scalability in term of dimensionality is crucial for any subspace clustering algorithm. As depicted in Figure 5.7(a), SUBCLU does not scale w.r.t. dimensionality. For the 20-dimensional data set the algorithm did not even finish after six days. The reason is the tremendous amount of 1 and 2-dimensional subspaces that have to be investigated with density-based clustering before processing any higher dimensional subspace. SCHISM as a grid-based approach has better runtimes, however, it is only an approximative technique as it loses density-based clusters due to its traditional grid structure. The EDSC algorithm overcomes the scalability problems of SUBCLU due to the efficient filter steps. And in contrast to SCHISM, the EDSC approach is lossless because of its novel density-conserving grid.

Scalability w.r.t. database size. In the next experiment we generate 15-dimensional data sets with different database sizes in the range of 500-5000 objects. As we can see in Figure 5.7(b), EDSC and SCHISM scale well w.r.t. the number of objects. SUBCLU also does not scale with increasing number of objects because of the time consuming database scans needed for density-based clustering. In contrast, EDSC



(a) MinPoints vs. runtime

(b) F parameter vs. runtime

(c) Gridsize vs. runtime

Figure 5.8: Parameter evaluation. Data: dimensions $d = 15$; database size $|DB| = 1500$

uses in its first filter step the novel density-conserving grid to avoid these scans and thus is nearly not influenced by the database size.

Selectivity. To illustrate the efficiency of the hypercube filter we analyze its selectivity compared to the number of density-based scans in SUBCLU. Recall that density-based clustering has quadratic complexity w.r.t. the number of objects. Thus avoiding density-based scans contributes significantly to the performance gains of EDSC. Figure 5.7(c) shows the number of required density-based clustering computations for the data from our first experiment with varying dimensionality. The highest dimensionality shown is 15-dimensional because SUBCLU does not scale to higher dimensional data. We see that indeed the main drawback of SUBCLU is the large number of regions that have to be clustered. Multi-step filtering in EDSC substantially lowers the number of density-based clustering computations. Many regions that have to be processed by SUBCLU are pruned by EDSC. Pruning is possible without any database scans in the first filter step, only based on the information of the grid cells and the novel border elements in the new density-conserving grid.

5.3.2 Parameterization

Detecting clusters in arbitrary subspaces may require tedious parameter tuning in some algorithms. We evaluate parameter setting for SUBCLU, SCHISM and EDSC

to study their parameter sensitivity and demonstrate that the EDSC algorithm is quite robust in terms of parameter settings.

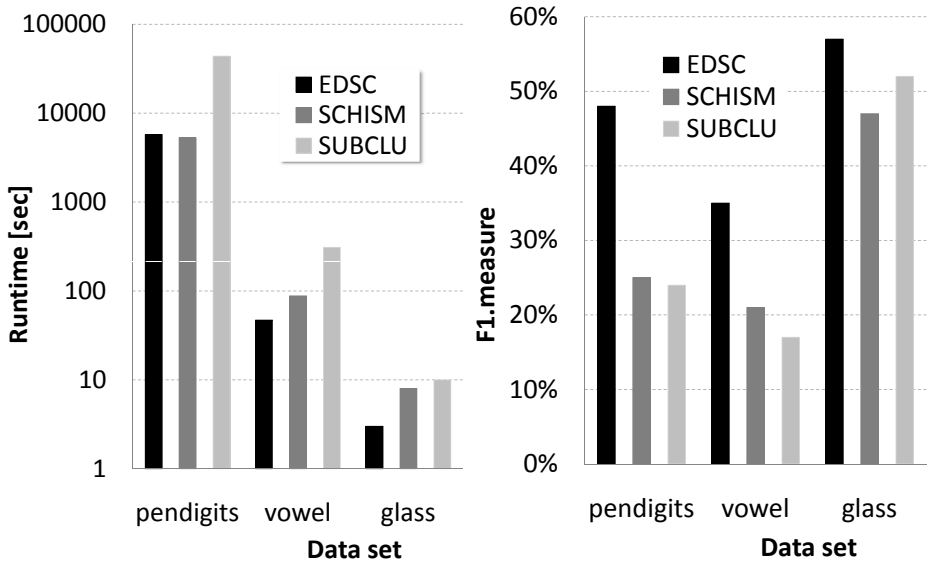
Fixed density thresholds. As already mentioned SUBCLU is dimensionality biased due to a fixed density threshold $MinPoints$. For different dimensional subspaces the measured density is not comparable. To find high dimensional subspace clusters one has to use a lower value for $MinPoints$ in SUBCLU. In EDSC this parameter is robust because of the unbiased density approach as proposed in Chapter 2. Figure 5.8(a) illustrates that with decreasing $MinPoints$ the runtime of SUBCLU increases. Finding high dimensional subspace clusters is virtually infeasible because already for $MinPoints = 8$ SUBCLU did not scale.

Variable density thresholds. As EDSC uses a variable threshold its density measure is comparable for clusters in different subspaces. In Figure 5.8(a) we see that the choice of $MinPoints$ has almost no effect on the runtime of the EDSC algorithm. As we can see in Figure 5.8(b) the second parameter F has almost constant runtimes, too. Using a unbiased density measure which automatically adapts to the dimensionality of the subspace, the variable density threshold is simply set with the fixed parameter F. F reflects the factor by which the variable expected density should be exceeded. Thus EDSC is easily parameterized by a constant and thus robust factor, yet adapts to the dimensionality of the subspace. Additionally the EDSC approach achieves a lossless pruning for this robust variable thresholds due to the second filter step based on the weak density criterion.

Gridsize. Grid-based algorithms like SCHISM suffer from sensitivity to the grid resolution. In Figure 5.8(c), runtime of SCHISM and EDSC for varying gridsizes are shown. The performance of SCHISM depends largely on the grid structure not only in terms of runtime as shown in Figure 5.8(c), but also in terms of accuracy (discussed later on, Fig. 5.9(b)). This is due to the sensitivity of SCHISM to grid cell resolution and positioning. EDSC is robust to positioning and resolution of the grid through its grid cell merges proposed in our novel density-conserving grid.

5.3.3 Quality and runtimes on real world data.

Class labeled data (pendigits, vowel and glass) from UCI machine learning repository [AN07] are used as ground truth to evaluate the quality of subspace clustering as in other recent subspace clustering approaches [AWY+99, SZ04]. The data cover a variety of dimensionalities and database sizes from 9 dimensions in glass, 10 in vowel up to 16 in pendigits, and with 214 objects in glass, 990 in vowel and 7494 in pendigits, respectively.



(a) Runtimes (b) Accuracy

Figure 5.9: Real world data

Runtimes for all data sets are given in Figure 5.9(a), the corresponding F1-values for accuracy measurements in Figure 5.9(b). From the F1-value results we can see that the enormous efficiency gains of the EDSC algorithm are achieved for an effective density-based model proposed in Chapter 2, that outperforms competing approaches. Thus, EDSC is an algorithm that due to its efficient but also lossless multi-step approach shows significantly better runtimes for a subspace clustering model of very high accuracy, in two datasets even better runtimes than the approximate SCHISM approach.

5.4 Efficient and lossless subspace clustering

We introduced EDSC, an efficient density-based subspace clustering algorithm. EDSC uses a database inspired multi-step approach which allows powerful pruning of the search space by exploiting two monotonicity properties. In our first filter step, EDSC efficiently prunes the search space without any density-based clustering. It is based on our novel density-conserving grid which ensures complete detection of density-connected regions by enclosing hypercubes. In our second filter step EDSC ensures lossless pruning for variable density thresholds by incorporating our weak density criterion. For both filter steps we prove lossless pruning and thus the completeness of our density-based subspace clustering. Overall EDSC achieves efficiency without jeopardizing accuracy, as our multi-step algorithm is capable of losslessly detecting subspace clusters with respect to unbiased density. Our experiments on large and high dimensional synthetic and real world data sets show that EDSC outperforms recent subspace clustering algorithms by orders of magnitude.

However, major challenges remain open even for the EDSC approach. The effi-

cient access to object counts in arbitrary subspace projections and the redundancy of subspace clusters are still not solved. Even though the grid-based solution reduces the computation cost significantly, we can additionally enhance efficiency by proposing an index support for subspace clustering. In addition to this index support we propose an in-process removal of redundancy in the following chapter. Thus, we tackle both efficiency challenges for the DUSC subspace clustering model.

Chapter 6

In-process removal of redundant subspace clusters

To detect clusters hidden in subspace projections of high dimensional data, subspace clustering focuses on relevant attribute projections for each individual cluster. As the number of possible projections is exponential in the number of dimensions, efficiency is crucial for high dimensional settings. Moreover, the resulting subspace clusters are often highly redundant, i.e. many clusters are detected multiply in several projections. Containing essentially the same information, redundant subspace clusters have to be removed to allow users to review the entire output. In addition, removal of low dimensional redundancy improves both the clustering quality and the runtime.

In this chapter we propose a novel index structure which enables in-process removal of redundant clusters in a novel processing schema. Unlike existing breadth-first approaches, INSCY (*IN*dexing *S*ubspace *C*lusters without redundanc*Y*) proceeds depth-first on the dimensionality of the subspaces. Our depth-first mining with index support allows immediate pruning of redundant subspace clusters and thus greatly reduces the computational cost of subspace clustering. Thorough experiments on real and synthetic data show that INSCY yields substantial efficiency and quality improvements over existing subspace clustering approaches.

6.1 Motivation and comparison to related work

Subspace clustering mines clusters in any possible subspace projection. As the number of possible subspace projections is exponential in the dimensionality of the data space, efficiency is a crucial issue in subspace clustering [AGGR98, KKK04]. Naively mining all possible subspace projections is clearly infeasible. The number of resulting subspace clusters is usually exponential as well. Clusters typically show up in different subspaces, generating redundant subspace clusters in low dimensional projections, which contain essentially the same information as the "maximal" high dimensional one. To allow users to analyze reasonable result sizes, redundant subspace clusters have to be removed. However, in typical breadth-first approaches, maximal high dimensional results are mined last. Their redundant low dimensional projections can only be cleared out once all redundant subspace clusters have been generated in a

costly process. Thus, in existing approaches redundancy yields overall low quality results but also poor runtimes.

Existing subspace clustering algorithms are based on the breadth-first processing schema proposed by CLIQUE and SUBCLU [AGGR98, KKK04]. Breadth-first subspace clustering algorithms are similar to the apriori algorithm originally introduced in frequent itemset mining [AS94]. They all work their way bottom-up on the subspace lattice. Starting from one dimensional results, all subspace clusters of dimensionality k are mined, before candidates of dimensionality $k+1$ are generated, and so on. For example, in Figure 6.1, all the subspaces clusters in the first ($\{1\}, \{2\}, \ldots$) and second ($\{1,2\}, \{1,3\}, \ldots$) level are mined before reaching the three-dimensional subspaces.

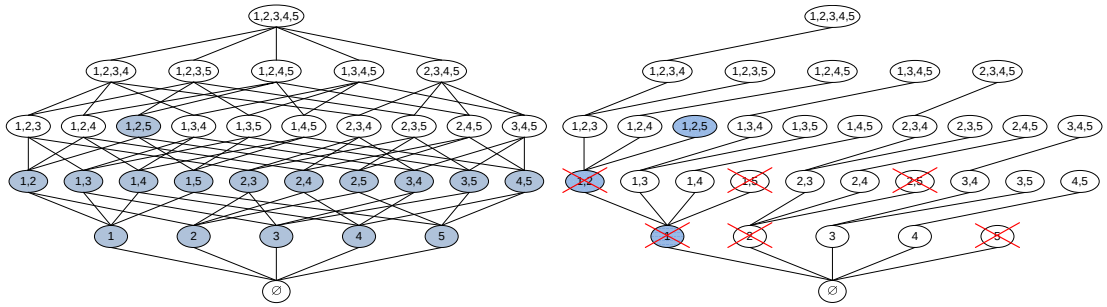


Figure 6.1: Breadth-first (left) vs. depth-first (right) processing of subspaces

Typically, the number of low dimensional subspace clusters is huge, as they reflect high dimensional subspace clusters in several projections. As in breadth-first approaches all low dimensional subspaces have to be processed before going into higher dimensional ones, these approaches have to perform costly density-based clustering on all lower dimensional projections even if they are all redundant. In breadth-first mining, redundant clusters can only be detected once the entire low dimensional projection has been processed. This means that no redundancy-based pruning is possible, as the redundant cluster is already output before its high dimensional representative is processed. Thus, these approaches show not only extremely large result sizes (redundancy) but also high runtimes (no in-process pruning of redundancy) of several days as stated for density-based subspace clustering in [KKK04].

Furthermore, as we know, the only type of index support to speed up such density computations in subspace clustering is the usage of inverted files for individual dimensions in [KKK04]. Indexing subspace clusters in breadth-first algorithms would require building index structures for each of the exponential many subspace combinations, which is clearly not advantageous.

In this work, we propose a novel depth-first processing with index support that allows for in-process-removal of redundant subspace clusters. As illustrated in Figure 6.1, our novel depth-first approach proceeds recursively by first processing higher dimensional cluster regions before proceeding to all of its low dimensional projections. This processing has two key advantages: First, as the maximal high dimensional projection is evaluated first, immediate pruning of all its redundant low dimensional

projections leads to major efficiency gains. Second, efficient mining support via indexing of potential subspace cluster regions is possible. Thus, depth first processing is the key to efficient index-based subspace clustering by in-process-removal of redundancy.

Our INSCY (*IN*dexing *S*ubspace *C*lusters with in-process-removal of redundanc*Y*) depth-first approach combines in-process redundancy pruning and index support to tap the full potential in subspace clustering algorithms. We introduce our new index structure, the *SCY*-tree, for subspace clustering to improve efficiency. The SCY-tree is a compact representation of potential subspace cluster regions which greatly reduces database scans. Our contributions thus include:

- depth-first subspace mining as novel processing schema
- automatic in-process-removal of redundancy
- a novel index for efficient subspace clustering

Substantial efficiency gains and automatically reduced output size render our INSCY approach fast and concise.

6.2 In-process removal of redundancy by depth-first processing

In general, redundancy is a major problem for subspace clustering approaches. As first method, INSCY proposes an in-process removal of redundant clusters to tackle this general challenge. In this section we first formalize our redundancy notion before describing our novel depth-first processing schema for in-process removal of redundant subspace clusters.

Redundancy in Subspace Clustering. Let us first illustrate redundancy in subspace clustering by a toy example as depicted in Figure 6.2. Assume the two-dimensional subspace cluster C_3 to be a valid density-based subspace cluster. It generates a redundant pattern C_2 when projected to the vertical axis, and a second redundant pattern C_1 when projected to the horizontal axis. Both C_1 and C_2 contain essentially less information than C_3 as the correlation of values in the two dimensions is not included. Typically, such redundant lower dimensional projections are output by existing methods but considered less informative by users.

To exclude such less informative subspace clusters we formalize this intuitive redundancy notion. Our redundancy definition follows the definition of non-redundant density-based subspace clusters as described in Chapter 2. We highlight the redundancy formalization of Definition 5 below. As redundancy is a general challenge for subspace clustering independent of the underlying cluster definition we skip the discussion of e.g. density-based clustering. One could have also taken other density notions as proposed in CLIQUE or SUBCLU [AGGR98, KKK04]. For simplicity of presentation, we use the DUSC definition with rectangle kernels for density

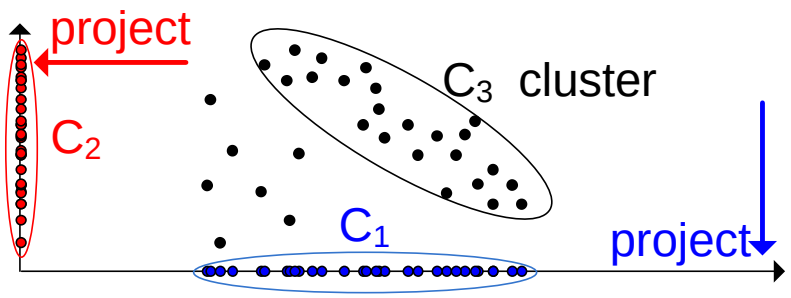


Figure 6.2: Redundancy in subspace clustering

assessment (i.e. counting the number of objects in the neighborhood, as e.g. in [EKSX96, KKK04]). We abstract from the adaptive density and focus only on the redundancy definition as the major part for this chapter.

Definition 21. Non-Redundant Subspace Clustering.
A subspace cluster (O, S) *w.r.t to Definition 5 is non-redundant iff:*

$$\neg\exists(O', S') \text{ subspace cluster with } O' \subseteq O \wedge S' \supset S \wedge |O'| \geq R \cdot |O|$$

Redundant subspace clusters are formalized as those subspace clusters O in subspace S that are already covered to a degree of redundancy R by a cluster $|O'| \geq R \cdot |O|$ in a higher dimensional subspace $S' \supset S$. We remove such redundant subspace clusters to allow users to analyze reasonable result sizes. With the parameter R one can tune the degree of acceptable redundancy in the range of $0 - 100\%$. With $R = 0\%$ no redundancy is accepted and thus redundancy is defined by the existance of a higher dimensional cluster. A cluster (O, S) is *redundant*: $\exists(O', S') : O' \subseteq O \wedge S' \supset S$. With $R = 100\%$ almost fully redundancy is accepted and thus a lower dimensional cluster is in the result as soon as it contains at least one object more than a higher dimensional cluster. A cluster (O, S) is *fully redundant*: $\exists(O', S') : O' \subseteq O \wedge S' \supset S \wedge |O'| = |O|$. Note that fully redundancy is not considered acceptable. A projection of a cluster with the same set of objects in a lower dimensional subspace contains essentially the same information and thus is not interesting at all.

Depth-first processing schema To exclude such a redundant cluster (O, S) one has to process all possible higher dimensional projections (O', S') with $O' \subseteq O \wedge S' \supset S$ first. Only by computing the highest dimensional cluster first one is able to exclude redundancy in-process without computing the redundant lower dimensional clusters. Obviously, such a processing requires a depth-first processing order of subspaces. By stepping recursively through the subspace lattice one can tackle the three major drawbacks of existing breadth-first approaches. First, depth-first does not produce large sets of candidates in low dimensional projections. Second, depth-first enables in-process-removal of redundant subspace clusters. And third, based on depth-first one enables possible index support to overcome repeated density computations in redundant candidate regions.

In the following, we show how incorporating redundancy removal into depth-first mining allows for more efficient and accurate subspace clustering. As main property in contrast to breadth-first subspace clustering, our algorithm does not need all lower dimensional subspaces for a restriction. In a recursive fashion on subspace regions, we first identify high dimensional subspace clusters and immediately prune all lower dimensional projections and thus avoid costly density computations on irrelevant low dimensional candidates.

Our algorithm thus processes each cluster region in all projections before proceeding to the next region. This processing is particularly advantageous as it allows for pruning redundant clusters during the mining process. In Figure 6.1 we assume a cluster found in subspace $\{1, 2, 5\}$. During the depth-first processing the shaded subspaces are recursively restricted to $\{1, 2, 5\}$. In contrast to breadth-first approaches $\{1\}, \{2\}, \{5\}, \{1, 2\}, \{1, 5\}$ and $\{2, 5\}$ do not have to be clustered to get to this subspace. If a cluster found in $\{1, 2, 5\}$ is the only non-redundant result, then we avoid the costly density computations for all of its lower dimensional subspaces. By our novel in-process removal we focus on new subspace clusters not yet found in higher dimensional projections. Technically, we step back in the recursion from high dimensional to lower dimensional projections, where all actual high dimensional clusters are available for immediate pruning of redundant regions in the crossed out subspaces in Figure 6.1. To overcome the costly database scans we propose a novel index structure in the following section. Based on the SCY-tree defined in Section 6.3.1, we propose our novel in-process pruning of redundant subspace clusters in Section 6.3.2. As we show in Section 6.3.3, this pruning is correct as it yields all and only non-redundant patterns.

Overall, by processing in depth-first order and using in-process redundancy removal we reduce costly density-based clustering and hence improve the overall efficiency of subspace clustering.

6.3 Indexing technique for subspace clusters

Our novel index for depth-first mining provides a compact representation of the given data with which we can access arbitrary subspaces without generating all lower dimensional projections. Thus in-process removal of redundant patterns is efficiently supported. To index subspace regions, we use a transformation from the original space to a compact tree structure. As the index is built only for clustering purposes it is only a temporary data structure. Investing a single database scan, the initial SCY-tree is built which represents the entire database in the full dimensional space. The SCY-tree structures the data such that each dimension is represented by one level of the tree, while the nodes on each level represent different regions in this dimension. After building the initial SCY-tree, further data access can be avoided as the general idea for our subspace processing on this tree structure is to recursively restrict the tree in arbitrary subspace regions without further database scans.

One restriction extracts the information of one region in one specific dimension. Thus, by recursively restricting more and more dimensions we implement a depth-first

search on the SCY-tree. With the first restriction the resulting tree represents a one dimensional subspace region, with every further restriction the resulting SCY-tree represents a path of (restricted) nodes out of the initial tree. For efficient mining without database access, subspace regions are described by these paths as they store at each node the count of objects in the respective subspace region. Consequently, by using this object count pruning of sparse regions and of redundant regions is performed directly on the SCY-trees without costly neighborhood computations. Only for validating cluster candidates, access to the original data is required. We describe our SCY-tree index before detailing its use in the INSCY algorithm.

6.3.1 SCY-tree structure

To efficiently support density-based subspace clustering by our novel indexing structure, we use a compact representation of potentially dense regions. Reconsidering Definition 5, we see that any potential cluster consists of a minimum number of dense objects, i.e. objects whose neighborhood count exceeds a certain threshold. We represent these more compactly by mapping regions from the original space to descriptor nodes that record the object count for these regions. Additionally, potential clusters are maximally S-connected, i.e. they consist of chains of dense objects within the ε-neighborhood of one another. To ensure that these connections are adequately reflected in the descriptor node representation, we additionally record information of objects within ε distance of any neighboring region.

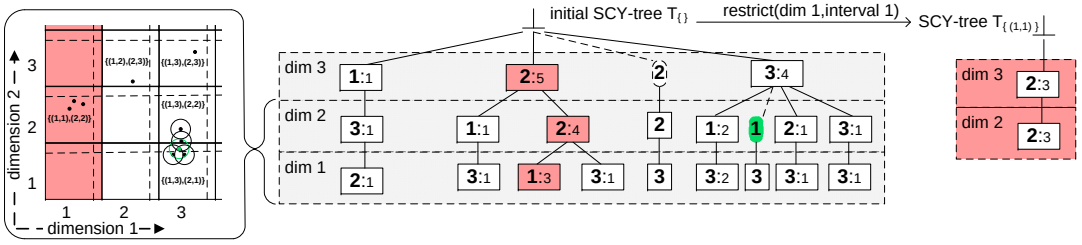


Figure 6.3: Initial SCY-tree and first recursion step

Each region can be described as the subspace dimensions it spans and the respective intervals in these dimensions. We base on the idea of our density-conserving grid as proposed in the previous chapter in Definition 19. Figure 6.3 illustrates[1] the basic idea: we use intervals in each dimension to describe regions in the original space and for each region we store the number of objects. The used intervals discretize the data space and allow efficient access to object counts in arbitrary subspace regions as aggregated information about the data distribution. For the discretization any grid-based representation could be used as our index is aware of density-based clusters spreading across two neighboring regions. Thus, we simply use equal width intervals as proposed in the CLIQUE approach [AGGR98]. An interval i in dimension

[1]For a better illustration we skipped the parent and link pointers as they are only necessary for algorithmic reasons.

d is specified by its lower $L_{(d,i)}$ and upper $U_{(d,i)}$ bound with width w such that an object o contributes to the count of this interval iff:

$$o_d \in (L_{(d,i)}, U_{(d,i)}] \text{ with } U_{(d,i)} = L_{(d,i)} + w$$

Additionally we ensure S-connectedness by regions of ε-width such that an object has to be present in such a region if two neighboring regions contain a S-connected cluster. Border objects in the density-conserving grid are represented by so called *S-connectors* in the SCY-tree. As depicted in Figure 6.3 the green S-connector node represents the green object at the border of interval 1 in dimension 2. Due to the properties of the density-conserving grid the S-connectors are used to ensure the detection of all density-based subspace clusters. Thus, in contrast to grid-based techniques like CLIQUE [AGGR98] we ensure to find density-based subspace clusters and thus we do not cut clusters apart. The discretization in fixed intervals might represent one density-based cluster in two or more intervals. By introducing additional ε-width intervals (S-connectors in the SCY-tree) at the borders of each interval we ensure to detect such a spread density-based cluster. An S-connector node for interval i in dimension d is specified by an ε-width area at the upper bound of this interval such that an object contributes to this indicator iff:

$$o_d \in (U_{(d,i)} - \varepsilon, U_{(d,i)}]$$

Thus, we are able to merge neighboring intervals and represent the set of S-connected objects in one SCY-tree. A required merge is simply indicated by an object in one of our additional S-connectors.

For the given high dimensional data we perform one database scan and create the initial SCY-tree on which mining operations are performed without excessive database scans. The initial SCY-tree is build by storing the object count of each grid region in a tree structure. Each full space region is represented by a path where nodes at level d represent the object count of this region in an interval i in dimension d. Paths representing only a subset of the dimensions (a subspace region) can be extracted out of this initial SCY-tree by a restriction operation. As depicted in Figure 6.3 for a 3-dimensional dataset we show one 2-dimensional projection of the data on the left side and the initial SCY-tree $T_{\{\}}$ in the middle. It represents the whole database, while a restricted SCY-tree $T_{\{(1,1)\}}$ on the right side represents all data objects that are part of region 1 in dimension 1. This tree is simply constructed by extracting the highlighted path out of the initial SCY-tree. It represents a single subspace region (interval 1 in dimension 1) and can be recursively restricted to higher dimensional projections of this region. As we can see, the tree contains a single path, as all three objects in this region are located in interval 2 in dimension 2 and in interval 2 in dimension 3. Thus, the only possible restriction leads to this higher dimensional subspace region $\{(1,1),(2,2),(3,2)\}$ specified by the highlighted path in the initial SCY-tree.

The general idea is that one can restrict a region to further dimensions by restricting the corresponding SCY-tree representing this region. Both regular intervals and our additional S-connectors are treated in the same way. The only difference

is that S-connectors do not store a count as they only indicate a required merge operation. How to restrict trees and how to handle density-based clusters across multiple regions via merging of trees is described in the next section.

Definition 22. SCY-tree structure
A SCY-tree T_D represents a region
$D = \{(d_1, i_1), \ldots, (d_k, i_k)\}$ in an arbitrary subspace.
The SCY-tree consists of nodes, each of them stores:

- *a descriptor (d, i) for dimension d and interval i of the region*
  regular nodes store the count c of objects in the respective region
  S-connector nodes store no count they indicate a required merge
- *a pointer to the parent node in the upper level of the tree and a list of child pointers to the lower level of the tree*
- *a pointer to a node with the same descriptor, providing a linked list of all nodes representing the same region in this dimension i.e. in this level of the tree*

Overall, the nodes of the SCY-tree are organized in levels where each level d stores the information of regions in dimension d.

The main properties of a node are the descriptor and count value. For shorter representation we thus address a node simply by $(d, i) : c$. The SCY-tree nodes are ordered lexicographically according to their descriptors: first, according to the dimensions and second according to the interval. We omit the dimension in the illustration of each node (cf. Figure 6.3) as each level of the tree corresponds to one dimension. The additional ε-width regions (*S-connectors*) that ensure correct density-based clustering are represented by $(d, i) : -1$ as we are only interested in the presence of an object. Nodes representing S-connectors are depicted without a count value (cf. green node in Figure 6.3) and only technically marked with a -1 count value to identify them in contrast to regular nodes.

By using S-connectors we ensure that each region containing a density-based cluster is represented by one SCY-tree. We therefore set-up the S-connectors at the upper border of each interval. Given an S-connected cluster spreading across two neighboring regions there has to be at least one object in this ε-width region. For the mining this indicates that INSCY has to merge these two neighboring regions to have the whole S-connected cluster represented in one SCY-tree. For a density-connected set of objects spreading over (d, i) and $(d, i+1)$ this means that a chain of S-connected objects with $o \in (d, i)$, $p \in (d, i+1)$ and $dist_S(o, p) \leq \varepsilon$ exists. Thus, object o has to be in the respective S-connector: $o_d \in (U_{(d,i)} - \varepsilon, U_{(d,i)}]$ indicating the merge operation. For multiple regions this merge operation on neighboring SCY-trees can be done iteratively until no further object is contained in any surrounding ε region. As discussed in Chapter 5, the underlying density-conserving grid structure ensures this lossless mining in a general filter and refinement architecture. Now in this chapter we have to additionally ensure that the lossless mining is efficiently supported by our novel index structure, which we will discuss in the following.

6.3.2 SCY-tree processing for non-redundant clustering

INSCY mines subspace clusters directly on SCY-tree paths that correspond to regions and their S-connected regions. We give an overview of INSCY in Algorithm 4, before providing details about each of the steps.

For each descriptor (line 1), i.e. region to be mined, the initial SCY-tree is restricted (line 2) to exactly the paths of this descriptor, such that the information for this region is compactly represented in the restricted SCY-tree. If the region has S-connected neighbors, as indicated by S-connector nodes with -1 counts, the corresponding trees are merged to generate a tree for the entire potential subspace cluster region (line 3). Sparse areas are removed simply by checking count values of tree paths for immediate pruning of unnecessary recursive calls (line 4). If a region cannot be pruned, it is further restricted recursively, i.e. higher dimensional projections are generated (line 5). If a high dimensional subspace cluster has been detected, redundant low dimensional projections are pruned directly (line 6). In the end, all non-redundant subspace clusters are output (line 7). Depending on the underlying subspace cluster definition, database access for density-based clustering in *DBClustering* is required. We use the DUSC model requiring a database scan for each candidate region to determine the density-connected subspace cluster.

Algorithm 4 INSCY(scy-tree, result)

```
1 foreach descriptor in scy-tree do
2     restricted-tree := restrict(scy-tree, descriptor);
3     restricted-tree := mergeWithNeighbors(restricted-tree);
4     pruneRecursion(restricted-tree); //prune sparse regions
5     INSCY(restricted-tree,result); //depth-first via recursion
6     pruneRedundancy(restricted-tree); //in-process-removal
7     result := DBClustering(restricted-tree) ∪ result;
```

For the detailed description, each step is followed by a running example, assuming parameter values $minSize = 4$; $R = 80\%$.

1. Restricting SCY-trees: searching different subspaces

In one recursive INSCY call the for-loop restricts the current SCY-tree for each descriptor (d, i) in lexicographical order of descriptors. Restriction means that only objects in dimension d in interval i contribute, consequently the tree decreases by at least level d. This step is similar to conditional FP-growth steps in association rule mining that builds conditional subtrees for frequency counts [HPY00]. While the restriction operation is similar to the FP-growth processing, the underlying structure of the SCY-tree provides additional information (not available in an FP-tree) for density-based subspace clustering. Thus, we require additional processing steps to extract this information as discussed in the following. To extract a restricted SCY-tree for interval i in dimension d one simply copies the paths from all nodes labeled with descriptor (d, i) up to the root node of the given SCY-tree T_D. These paths are inserted into a new SCY-tree $T_{D\times(d,i)}$ representing subspace region D restricted in

(d, i). Note that any descriptor, not only those at leaf level, are picked for restriction. This is crucial for detecting arbitrary subspace clusters in any possible combination of dimensions. INSCY can thus efficiently obtain any possible subspace by extracting the respective paths into a restricted tree. For efficient restriction, all nodes labeled with the same descriptor are accessed via a linked list.

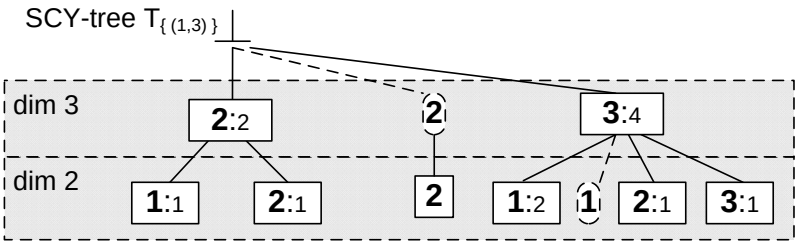


Figure 6.4: SCY-tree for dimension 1 interval 3

Example. The initial SCY-tree in Figure 6.3 is first restricted to $(1, 1)$ following lexicographical order, i.e. in dimension 1, interval 1 (red region). For each node in dimension 1 (at leaf level) labeled 1 (only the third leaf node from the left with count 3), the paths from this node to the root are copied into the restricted SCY-tree $T_{\{(1,1)\}}$, depicted at the right, labeled with the count of that node. In further recursive steps this SCY-tree could be restricted to $T_{\{(1,1)\times(2,2)\}}$, $T_{\{(1,1)\times(2,2)\times(3,2)\}}$ and $T_{\{(1,1)\times(3,2)\}}$ which represent all possible subspace projections for these three objects. Designed as a depth-first search, we thus would extract all possible subspace regions out of the initial SCY-tree.

2. Pruning recursive calls: removing sparse regions

However, if any SCY-tree count does not exceed the parameter $minSize$, further restriction, i.e. projection, can only lead to lower count values that do not exceed $minSize$ either. Consequently, they can be safely pruned.

Example. There is no S-connector for $T_{\{(1,1)\}}$ in dimension 1, thus the region cannot be grown by merge operations. The count of 3 is below our $minSize$ value of 4, thus the tree is pruned.

3. Merging SCY-trees: growing S-connected regions

Potentially S-connected restricted trees are indicated by S-connector nodes $(d, i) : -1$. Such a node indicates a merge of (d, i) and $(d, i + 1)$ as an object is present in the ε-width region at the border of these two neighboring intervals. The respective SCY-trees $T_{D\times(d,i)}$ and $T_{D\times(d,i+1)}$ have to be merged as an S-connector $(d, i) : -1$ in T_D indicates a cluster spreading over both subspace regions. Merging of S-connected restricted SCY-trees representing these two neighboring subspace regions means simply inserting all paths of one tree into the other one, aggregating the count values on common paths, possibly inserting new nodes.

Example. Restricting the SCY-tree $T_{\{(1,3)\}}$ (Fig. 6.4) in dimension 2 we get the two SCY-trees for $(2, 1)$ and $(2, 2)$ shown in Figure 6.5. Neither exceeds $minSize$,

but the S-connector node 1 in dimension 2 in the parent tree $T_{\{(1,3)\}}$ indicates a connection in this dimension from $(2,1)$ to the lexicographical neighbor $(2,2)$. The two SCY-trees are merged to the SCY-tree $T_{\{(1,3)\times(2,1-2)\}}$ representing both intervals in dimension 2, depicted in Figure 6.5.

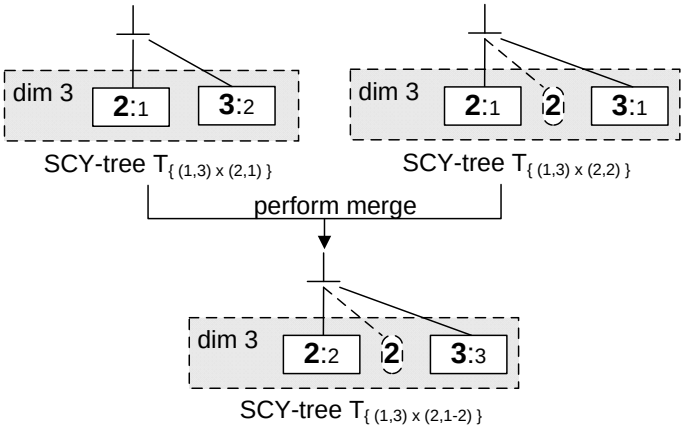


Figure 6.5: Merge of two SCY-trees

4. Clustering: mining actual subspace clusters

In the final step, the density-based clustering on the restricted SCY-tree, the actual data is accessed to identify the actual neighborhoods and to check all conditions of subspace clusters (Definition 5). This access, however, is required only for those few regions that cannot be pruned.

Example. Figure 6.5: $T_{\{(1,3)\times(2,1-2)\}}$ is merged in dimension 3 according to S-connector node 2, yielding $T_{\{(1,3)\times(2,1-2)\times(3,2-3)\}}$ with a count of $5 \geq minSize$. There is no further dimension that we could restrict the tree in. Since our result set is still empty, no redundancy pruning is possible, and we cluster the region $(1,3) \times (2,1-2) \times (3,2-3)$. In our example, we assume detection of a cluster.

5. Redundancy pruning: in-process removal

In-process-removal of redundant clusters is checked on the SCY-tree (with respect to the redundancy parameter R) if there is already a cluster in the result set covering this region. As INSCY proceeds in a depth-first manner, the recursion stops if no higher dimensional clusters can be found (i.e. no further restriction is possible, or $minSize$ is no longer exceeded). INSCY then steps back to analyze lower dimensional projections, checking only SCY-trees containing potentially non-redundant clusters. As the maximal high dimensional subspace clusters is the first one to be included in the result set, costly mining steps of low dimensional projections are avoided.

Example. Stepping back in the recursion, process $T_{\{(1,3)\times(2,1-2)\}}$. The redundancy check shows that SCY-tree $T_{\{(1,3)\times(2,1-2)\}}$ has a count of only 5. Thus no DBClustering is performed, because this is exactly the size of the subspace cluster found in the 3-dimensional subspace $(1,3)(2,1-2)(3,2-3)$.

6. Arbitrary restrictions: detecting all subspace clusters

The INSCY algorithm mines all subspaces, i.e. any possible combination of dimensions. Any dimension not currently under consideration, be it at leaf level or above, is simply disregarded during restriction of SCY-trees. More precisely, restriction starts at the leaf level and first of all considers all dimensions for restriction of the highest possible subspaces in depth-first order. However, during depth-first search it may skip some nodes, i.e. if none of the nodes in dimension d are used for restriction this means that this dimension is ignored in the considered subspaces. During depth-first processing arbitrary dimensions are skipped and consequently, SCY-trees provide the means for efficient mining of any subspace.

Example. Another step back in the recursion leads to $T_{\{(1,3)\}}$ shown in Figure 6.4. Here, dimension 2 has already been processed, and it does not need to be restricted any more. Next, restriction in dimension 3 is mined. From Figure 6.4 we can see that the resulting SCY-tree $T_{\{(1,3)\times(3,2-3)\}}$ would have a count of 6, as there is a count of 2 in cell 2 and a count of 4 in cell 3 with an S-connector path between them. Redundancy pruning is possible again, as with a redundancy parameter of 80% the inequality ($5 \geq 80\% \cdot 6$) is true and thus a count of 6 is considered too close to the already detected subspace cluster of size 5 for mining or inclusion in the result set. In the last step back to the initial SCY-tree, the INSCY algorithm detects a non-redundant cluster. Skipping dimension 1 and performing restriction and merge for dimension 2 yields $T_{\{(2,1-2)\}}$ with a count of 8. A further recursive call of INSCY on dimension 3 performs the second DBClustering call on $T_{\{(2,1-2)\times(3,2-3)\}}$ because this time ($5 \geq 80\% \cdot 8$) does not hold for redundancy pruning.

6.3.3 Proof: All and only non-redundant subspace clusters

In this section, we proof the correctness of in-process-removal of redundant clusters by showing that INSCY detects all and only non-redundant subspace clusters. Our approach ensures that, first, all non-redundant subspace clusters are in the result listing and that, second, there are only non-redundant subspace clusters in the listing. Let us recall that a cluster is non-redundant if the objects have not been found in a higher dimensional cluster with respect to the redundancy parameter R in Definition 21.

The first aspect that all non-redundant clusters are in the listing is straightforward as any SCY-tree that is pruned with respect to redundancy is compared against subspace clusters that have already been stored in the result listing. As no subspace cluster is later removed from the listing, all non-redundant clusters are included.

What remains to be shown is that the listing does not contain any redundant subspace clusters. We show this fact by analyzing the processing order. INSCY mines regions in a depth-first manner, i.e. each individual region is clustered in all subspace projections. Due to the recursive nature of the algorithm, higher dimensional subspace clusters are mined first, before stepping back to possible non-redundant lower dimensional subspace clusters in the same region.

Theorem 7. In-process removal of redundancy.
INSCY mines exactly the non-redundant subspace clusters.

Proof. Assume to the contrary that the INSCY algorithm adds a redundant subspace cluster (O_R, S_R) to its result listing. As subspace cluster (O_R, S_R) is redundant, there is a higher dimensional subspace cluster (O_H, S_H) with $O_H \subseteq O_R \wedge S_H \supset S_R \wedge |O_H| \geq R \cdot |O_R|$. We show that this assumption cannot be true by studying two cases:

- case 1: $((O_H, S_H)$ is mined before $(O_R, S_R))$
  As we have (O_H, S_H) already in the result list, INSCY detects the redundancy of (O_R, S_R) (checking the redundancy condition in Definition 21) before performing costly density-based clustering. Thus, (O_R, S_R) is not inserted into the result list.

- case 2: $((O_H, S_H)$ is mined after $(O_R, S_R))$
  As (O_H, S_H) is a higher dimensional subspace cluster projection of (O_R, S_R), the subspace S_H consists of all dimensions that are relevant for subspace cluster (O_R, S_R): $S_H \supset S_R$. Due to the processing order of our recursive depth-first search, INSCY had to further restrict the subspace S_R to S_H before processing S_R with DBClustering. Thus case 2 cannot occur.

This concludes the correctness proof. □

As we have shown, INSCY mines all non-redundant subspace clusters with respect to Definition 21. Furthermore, we show in the following section that the removal of redundancy yields not just better runtimes but also better quality. This can be explained by the fact that typically the higher dimensional subspace clusters best describe the data objects.

6.4 Experiments

We demonstrate the performance of our INSCY approach in terms of efficiency as well as in terms of successful removal of redundant subspace clusters.

INSCY is compared to the breadth-first algorithm SUBCLU [KKK04]. It is the most recent approach to detect density-based subspace clustering without approximation. It uses inverted files on individual dimensions to enhance neighborhood queries for density computation, since other indexing techniques are not beneficial for breadth-first subspace clustering. To the best of our knowledge, there is no other indexing approach for subspace clustering.

6.4.1 Synthetic data

We generate synthetic data for scalability experiments following a method proposed in [KKK04] to generate density-based clusters in arbitrary subspaces. Given the subspace and the number of objects for each cluster, the generator creates dense regions separated by noisy regions. In addition, our generator takes into account

that objects can belong to multiple subspace clusters, just as in most real world data sets. We generate data of different dimensionalities and hide subspace clusters with a maximal dimensionality of 80% of the data dimensionality. Subspace clusters are hidden in the synthetic data with partial overlapping of objects.

Redundancy removal. We first demonstrate the effects of in-process-removal of redundancy. Figure 6.6 illustrates the effect of varying the degree of redundancy R for subspace clusters (see subspace cluster Definition 21) for different database sizes. The higher R, the more redundant subspace clusters are included in the result set. As we can see, performance depends largely on the degree of redundancy. No redundancy, i.e. $R = 0\%$ shows runtimes far lower than all other variants. Even as little as $R = 20\%$ redundancy in the result set leads to four times slower runtimes (depending on the database size). Towards full redundancy of $R = 100\%$ exponential runtime behavior is only slightly better than SUBCLU. Thus, redundancy removal is clearly important for runtime performance. As shown in the next experiment, redundancy removal is also highly beneficial for quality of the result.

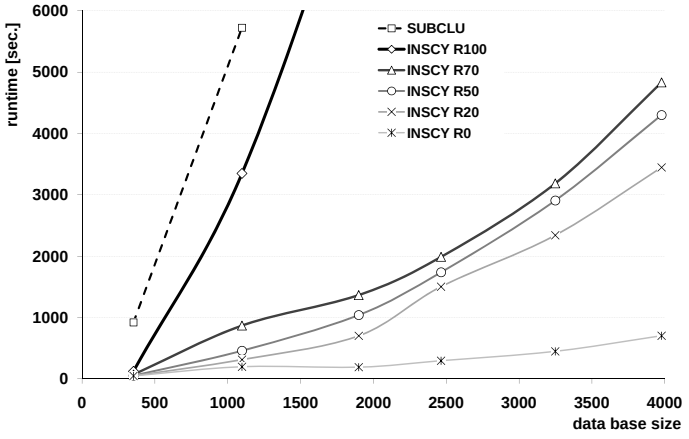


Figure 6.6: Redundancy

For quality measurements we use the $F1$ value that is commonly used in evaluation of classifiers and recently also for subspace or projected clustering as well [WF05, MSE06]. It is computed as the harmonic mean of recall ("are all clusters detected?") and precision ("are the clusters accurately detected?") values, respectively. The F1-value of the whole clustering is simply the average of all F1-values. The class label assigned to any detected subspace cluster is its most frequent class label.

Figure 6.7 illustrates F1 quality measurements and the size of the output for the same experiment as before (database size = 1900 objects). As we can see from the dark gray bars, the number of subspace clusters increases exponentially (note that the scale to the right is logarithmic). Thus, users would be overwhelmed by the output size of competing algorithms. The light gray bars indicate the F1 quality measurements. Clearly, removing redundancy leads to far better quality as F1 rates the detection of all and only hidden clusters as the optimal solution. Redundant subspace clusters obscure the information in maximal dimensionality subspace clusters. Thus,

removing redundancy yields better efficiency and effectiveness in subspace clustering. If no redundancy is removed like in SUBCLU the number of subspace clusters is tremendous. The SUBCLU algorithm did not finish after more than 10 days. Hence we were not able to evaluate the F1 value for SUBCLU.

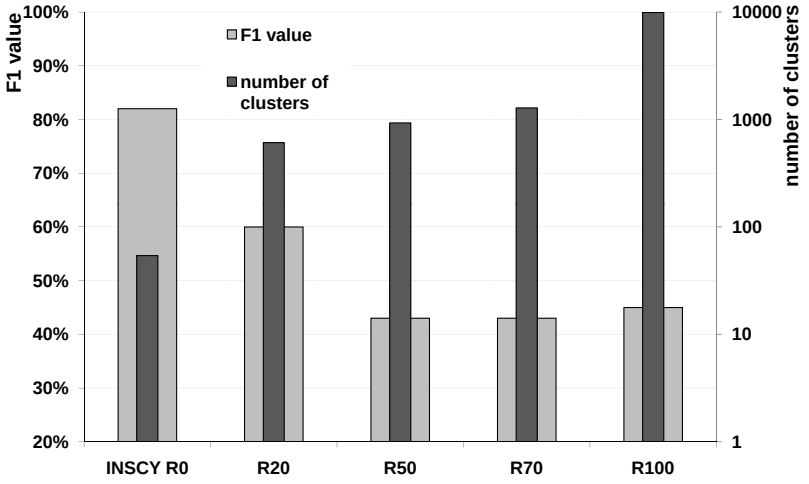


Figure 6.7: Redundancy Quality

6.4.2 Real world data

We analyze both quality and efficiency of INSCY subspace clustering on four real world data sets, which are summarized in Table 6.1. To measure $F1$ values on real world data where no ground truth on the number and size of subspace clusters is known, we use class label information in the data as ground truth.

	Objects	Dimensions	Classes	Source
Pendigits	7494	16	10	[AN07]
Vowel	990	10	11	[AN07]
Glass	214	9	6	[AN07]
Shapes	160	17	9	[KWX+06]

Table 6.1: Real world data sets

For each data set $F1$ quality measurements are given in Figure 6.8 for SUBCLU and for INSCY with three different redundancy settings, $R = 0\%, 2\%, 5\%$. The bars nicely illustrate that removing redundancy is important for all four real world data sets as well. This is an interesting result which indicates that the redundancy notion in Definition 21 indeed removes irrelevant clusters which are blurred by noise. If clusters which contain noise are removed from the result set, the precision is improved. For example using $R = 0\%$ results in a precision of 30% for the Vowel data while using $R = 5\%$ yields a precision of 24%. Removing all redundant subspace clusters might miss a few cluster objects resulting in slightly lower recall. As illustrated by Figure 6.8

allowing a very small amount of redundant clusters is a good compromise between a high precision and recall. Consequently, removing redundancy from real world subspace clusterings is an important step not only for allowing users to inspect the result but also in improving the quality.

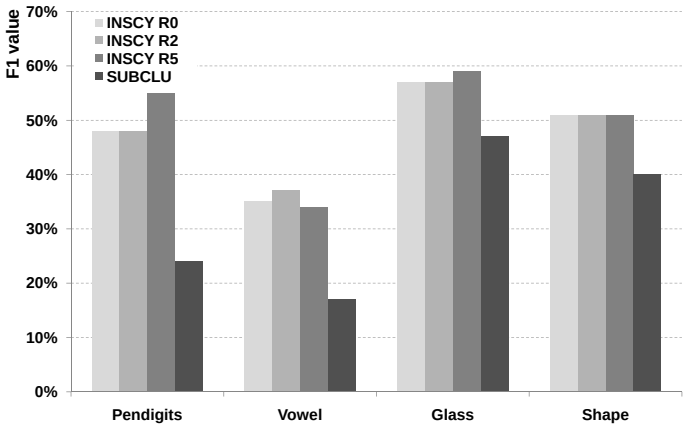


Figure 6.8: Quality on real data

As discussed, our proposed INSCY algorithm is the first algorithm which is able to speed up subspace clustering by in-process-removal of redundant clusters. The effect for runtime behavior is presented in Figure 6.9. The results demonstrate that INSCY outperforms SUBCLU, especially for settings which exploit the full potential of redundancy pruning ($R = 0\%$).

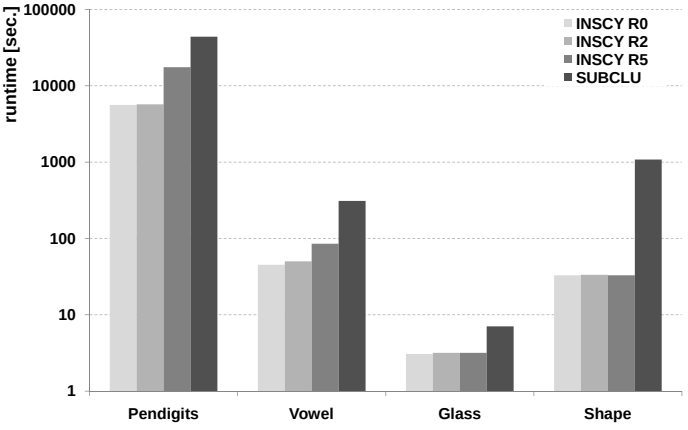


Figure 6.9: Runtime on real data

Summing up, INSCY indeed outperforms existing subspace clustering approaches on both synthetic and real world data. In-process-removal of redundant subspace clusters in our depth-first index supported algorithm yields superior runtime performance while actually improving the quality of the result and reducing the output to a manageable size.

6.5 Enhancements by in-process removal of redundancy

We introduced the INSCY (indexing subspace clusters with in-process removal of redundancy) approach. It overcomes two major drawbacks of existing subspace clustering approaches: highly redundant mining results (cf. Challenge 3) and poor runtimes of previous processing schemes (cf. Challenge 4). We analyzed how in-process removal of redundancy in a depth-first approach permits powerful pruning of the subspace cluster search space. Our SCY-tree is the first indexing structure for compact representation of potential subspace cluster regions. Our INSCY algorithm avoids repeated database scans by building compact SCY-trees. Thorough experiments demonstrate that INSCY clearly outperforms existing subspace clustering algorithms while automatically reducing the output to non-redundant information.

While INSCY is the first efficient method proposed for non-redundant subspace clustering, we extended this fundamental idea of indexing subspace regions to several other techniques. We describe further enhancements of our SCY-tree structure to heterogeneous data in the following chapter. Especially we propose a unification of density-based subspace clustering and frequent itemset mining for efficient subspace mining on heterogeneous data.

Chapter 7

Unification of subspace clustering and frequent itemset mining

Heterogeneous data, i.e. data with both categorical and continuous values, is common in many databases. However, most data mining algorithms assume either continuous or categorical attributes, but not both. In high dimensional data, phenomena due to the "curse of dimensionality" pose additional challenges. Usually, due to locally varying relevance of attributes, patterns do not show across the full set of attributes. While for continuous valued attributes we have shown that subspace clustering can efficiently detect dense subspace clusters, categorical attributes show different characteristics and it is not meaningful to mine categorical data with our techniques proposed in the previous chapters. Especially as frequent itemset mining provides a very efficient paradigm for mining binary and categorical data. Thus, a unification of both subspace clustering and frequent itemset mining would be of great importance for an overall efficient computation of both continuous and categorical attributes.

In this chapter we propose HSM, which first defines a unified pattern model and second proposes an efficiency enhancement for heterogeneous high dimensional data. It allows data mining in arbitrary subsets of both categorical and continuous attributes. Based on the unified HSM model we propose an efficient algorithm, which is aware of the heterogeneity of the attributes. Thus, we incorporate our efficient subspace clustering with the efficiency enhancements of frequent itemset mining. We extend our SCY-tree indexing structure for continuous attributes such that HSM indexing adapts to different attribute types. In our experiments we show that HSM efficiently mines patterns in arbitrary subspaces of heterogeneous high dimensional data.

7.1 Introduction to heterogeneous subspace mining

In this work, we focus on mining of heterogeneous data types with continuous and categorical attributes, as found in many applications. The discovery of meaningful patterns that provide a consistent view on both data types requires a pattern model that is consistent in defining interestingness and novelty for continuous and

categorical attributes alike.

We propose a novel model for heterogeneous data mining in high dimensional data. Our model provides a consistent view on both data types, and derives a common subspace clustering notion. Based on previous work on density-based subspace clustering in continuous data (cf. Chapter 2), we extend our model in this work to heterogeneous data. We highlight common challenges in subspace clustering and frequent itemset mining by giving a thorough review and comparison of models in both areas, before we present our novel model covering both areas. Our consistent model ensures comparable subspace clustering results on heterogeneous data by a consistent normalization of both density and frequency measures.

As not only our recent research has shown, pattern detection in projections of high dimensional data is a computationally challenging task [AGGR98, KKK04]. Especially for different data types, we need adaptive algorithms which ensure efficient pattern detection in projections of heterogeneous data. We propose an algorithm for heterogeneous subspace clustering that exploits the nature of continuous and categorical attributes in a novel index structure. This index is based on our previous work on efficient density-based subspace clustering for continuous data as descibed in Chapter 6. We extend this SCY-tree for continuous attributes in this work to an adaptive HSM-tree index structure for heterogeneous data, by extracting common mining characteristics of different attribute types in subspace clustering and frequent itemset mining. Although it seems to be a simple combination of known indexing techniques, the novel automatic adaptation between continuous and categorical attributes is essential for efficient mining on heterogeneous data. As we show in thorough evaluations our algorithm achieves a significant efficiency improvement compared to recent subspace clustering algorithms.

	d_1 noise level	d_2 temperature	d_3 humidity	d_4 illumination	d_5 vegetation	d_6 animals
o_1	3 dB	15°C	54%	20000 lx	FOREST	MAMMALS
o_2	7 dB	13°C	53%	70000 lx	FOREST	MAMMALS
o_3	5 dB	12°C	58%	100 lx	FOREST	MAMMALS
o_4	7 dB	10°C	56%	80 lx	FOREST	MAMMALS
o_5	8 dB	9°C	54%	20 lx	FOREST	MAMMALS
o_6	6 dB	9°C	52%	1 lx	FOREST	MAMMALS
o_7	3 dB	8°C	53%	0,0001 lx	FOREST	MAMMALS
o_8	4 dB	12°C	17%	15000 lx	GRASSLAND	INSECTS
o_9	5 dB	11°C	13%	60000 lx	GRASSLAND	INSECTS
o_{10}	5 dB	10°C	15%	10 lx	GRASSLAND	INSECTS
o_{11}	7 dB	8°C	14%	5 lx	GRASSLAND	INSECTS
o_{12}	8 dB	9°C	13%	0,01 lx	GRASSLAND	INSECTS
o_{13}	4 dB	42°C	1%	85000 lx	DESERT	REPTILES
o_{14}	5 dB	45°C	3%	120000 lx	DESERT	REPTILES
o_{15}	6 dB	43°C	2%	90000 lx	DESERT	REPTILES
o_{16}	3 dB	46°C	3%	4000 lx	DESERT	REPTILES
o_{17}	2 dB	44°C	1%	0,01 lx	DESERT	REPTILES
o_{18}	6 dB	43°C	2%	10 lx	DESERT	INSECTS

Figure 7.1: Heterogeneous data running example

As a running example to illustrate our work, we use the toy data set given in Figure 7.1. Our example is based on a real world scenario: A sensor network measuring different types of environmental parameters, vegetation and prevailing animal types around the sensor. We have given a data set of 18 objects (sensor measurements) described by 6 dimensions (sensor types). There are four continuous data types: noise level, temperature, humidity, and illumination, as well as two categorical data types: vegetation and animals. We have highlighted the hidden patterns in the data by surrounding lines. For example we see that there are three frequent itemsets present in the categorical attributes. Each of them specifies a group of objects: $o_1 - o_7$, $o_8 - o_{12}$ and $o_{13} - o_{18}$. Taking also the continuous valued attributes into account one can observe the same grouping for the humidity attribute, while noise and temperature form other groupings. However, groups do not appear in all attributes, as the attribute illumination is noisy and provides almost random measurements. The goal is thus to identify a grouping of objects that are similar in a subspace of the high dimensional space, i.e. to identify the relevant subset of dimensions on which the grouping appears.

7.2 Paradigms for mining of heterogeneous data

To the best of our knowledge there is no technique that can detect heterogeneous subspace patterns in high dimensional data. In this section, we discuss existing approaches to mining of novel and interesting patterns in continuous or categorical domains. We analyze their advantages and disadvantages with respect to mining heterogeneous data and propose a consistent model for both data types. We illustrate our discussion with our running example.

7.2.1 Frequent itemset mining

For categorical attributes there exist algorithms for mining frequent patterns out of transaction data. A transaction data set consists of sets of items that e.g. in the typical market example are goods that were bought together. A frequent itemset is a combination of several items that were surprisingly often bought together. Each item is given by a categorical value.

Compared to our high dimensional data mining task there are two differences. First, each object in our case has a fixed number of given attributes, while transactions can be of different size. Second, attribute values in the same dimension can not occur together as description of an object, while items can be combined with arbitrary other items in transactions. For both differences we observe that our high dimensional case is a specialization of itemset mining. In addition, both mining tasks have a major point in common. Both have to identify a subset of relevant items (attribute value combinations). Thus, they are not interested in a partitioning of data but in the detection of multiple overlapping patterns per object. This common overlap is given by their simple definitions, as they output all dense or frequent patterns independent of how many of these patterns are detected for each object.

Formally, the frequent itemset mining definition mines for categorical data the objects that have exactly the same attribute values in a selected subspace. By counting the frequency of an itemset we thus measure the number of similar objects. This frequency measure is called the support of an itemset. An itemset has then to fulfill a given parameter value of $minSupport$ to be frequent.

The frequency measure and its properties are a major topic concerning the efficiency of mining. As a naive approach one would calculate the frequency of all possible item combinations, which is exponential in the number of different items occurring in the data. This is clearly inefficient. A first efficient approach is the Apriori algorithm [AS94]. It uses a monotonicity property on the frequency measure to efficiently prune the exponential search space. Monotonicity means that removing one item from an itemset P, the support of this reduced P' can only be greater or equal to the support of P. By removing an item one relaxes the condition for the frequency and thus the number of objects that support this pattern cannot decrease.

Definition 23. Monotonicity Property
Given a frequent itemset $P := \{I_1, \ldots I_k\}$, then all its subsets $P' \subset P$ are also frequent itemsets.

The Apriori algorithm first mines the frequent 1-itemsets and then iteratively combines bottom-up k-itemsets to $(k+1)$-itemsets using the monotonicity property to prune non-frequent $(k+1)$-itemsets. More efficient algorithms like the FP-growth algorithm proceed recursively and thus achieve efficient frequent itemset detection without the need of generating all smaller k-itemset candidates [HPY00]. Frequent itemset mining is able to mine groups of objects which are identical in a subspace of the high dimensional data. Using a monotonicity property on the frequency measure one can efficiently prune the exponential search space. In addition, indexing the data in the FP-tree provides an efficient detection of frequent itemset without candidate generation. However, the FP-growth algorithm is only applicable to categorical data.

7.2.2 Density-based subspace clustering

For continuous valued attributes it is not meaningful to count the frequencies of attribute values co-occurring in the database. As one has infinitely many possible values per attribute, the probability of finding two data items having the same value is low. Instead a range of values is usually examined. Given the natural order of real values one can specify a neighborhood around an object O. Objects inside this neighborhood can be seen as similar values while objects outside the neighborhood are dissimilar to the object O. This "density-based" paradigm detects arbitrary shaped clusters in continuous data [EKSX96]. Density-based clustering has been shown to successfully identify clusters of arbitrary shape even in noisy settings, while discretized and categorical data types conflict with the definition of dense areas in continuous spaces. The density-based paradigm defines clusters as maximal sets of *density-connected* objects. An object O is *dense* if its neighborhood, specified by an ε-radius around O, contains more than a threshold many objects.

Extending this paradigm to high dimensional data one is interested in dense subspace regions [KKK04]. However, for density-based clustering in different subspaces one has to cope with incomparable densities. In our previous work, we have defined an unbiased density-based subspace clustering for continuous valued data (cf. Chapter 2). The key property is to normalize the density by the expected density of the subspace dimensionality. The expected density is simply the average number of objects in the ε-neighborhood w.r.t. the dimensionality. This can be computed as the ratio of the volume of the ε-sphere to the volume of the subspace. In the following, we base on our unbiased density-based subspace clustering as given in Definition 5 and extend its density measure to heterogeneous data.

In general, the notion of density and density-connectivity is widely used for continuous valued attributes, especially to detect arbitrarily shaped clusters and to achieve a robust behavior in noisy settings. However, due to the density measure it is only applicable to continuous valued attributes. For density computation, database scans are needed which results in high computation costs, especially when compared with the highly efficient frequent itemset mining without candidate generation.

7.2.3 Comparison of mining paradigms

Our main aim in this work is to consistently extend the main properties described in our unbiased subspace cluster definition, for heterogeneous data. For categorical attributes a minimum number of occurrences in the data set makes sense. In frequent itemset mining it corresponds to the $minSupport$ a pattern has to fulfill. However, as we mine patterns in arbitrary subspaces, this minimum number has to be normalized according to expected frequency in this subspace. We use a normalized density measure for unbiased density-based subspace clustering to achieve a consistent cluster definition across arbitrary subspaces. For an overall consistent model on heterogeneous data we will extend this model by an unbiased frequency measure in the next section.

Redundancy of subspace clusters as proposed in Chapter 6 can be found in frequent itemset mining as well. It is called closed itemset mining, as one searches for the maximal number of co-occurring items [Zak00, PHM00]. As given by the monotonicity property (cf. Definition 23), all subsets of a frequent itemset are also frequent. The problem in both worlds, frequent itemset mining and subspace clustering, is that one is not interested in such obvious redundant information. The main difference between the two models is the density and frequency notion, as both are related to characteristics of continuous or categorical values. Density connectivity and maximality as given in Definition 5 are not directly applicable for categorical data. To achieve an overall consistent model we have to be aware of these characteristics. Each of them is meaningful in its domain. Thus, our main aim is to ensure these differences without mixing up continuous and categorical attributes by transformation or discretization.

A naive way of integrating both attribute types could be discretization of the continuous attributes, e.g. by grid-based discretization used for subspace clustering [AGGR98]. However, discretization loses some knowledge about the data, and thus

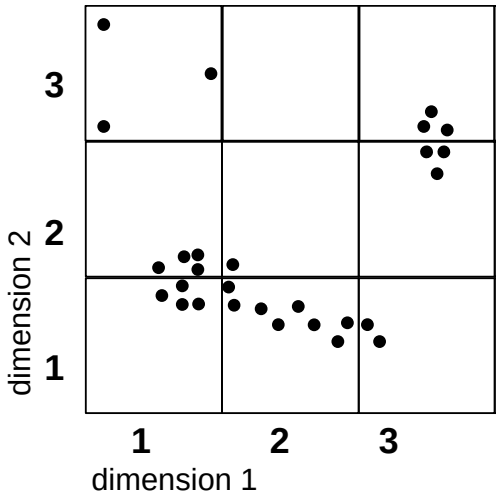


Figure 7.2: Grid based discretization

lacks a main property of density-based clustering. Density-based clustering is capable of identifying arbitrarily shaped clusters by taking the neighborhood around objects into account. Grid-based clustering approaches, however, can only detect clusters that consist of cells with more objects than the specified threshold. Consequently, the clustering results of these algorithms are heavily sensitive to the grid resolution and position. Figure 7.2 illustrates this problem: assuming a threshold of at least five objects, the only cluster that would be detected is the one located in cell (2,1). Moreover, this cluster would be revealed only partially, as the remaining parts of the cluster do not exceed the threshold of five objects in their respective cells. Additional post-processing of such clusters could remedy this problem, yet a cluster that is cut apart by the grid such that none of these partitions exceed the threshold (e.g. in cell (3,2) and cell (3,3)) would still be missed completely. If one were to lower the threshold to just three objects, this cluster would be detected, yet more false hits would be produced (e.g. cell (1,3) would produce a pseudo-cluster). The finer grained grid cells are, the less likely distortions are, yet the runtime complexity increases accordingly. Only density-based clustering is capable of detecting all clusters reliably.

7.3 HSM pattern model

For categorical data, frequency counting is more meaningful and efficient than the complex density measures, while for continuous data density-based clustering can detect arbitrary shaped clusters and is robust against noise. A combination of both types of models is the goal. We achieve this in an overall consistent manner by incorporating both types of measurements into our HSM pattern model.

In density-based (subspace) clustering, the density of an object O is measured via a density measure $\varphi(O)$. This can be simply the number of objects in the ε-neighborhood of O ($\varphi(O) := |N_\varepsilon^S(O)|$) or any other density measure as proposed in Chapter 2. Objects in the neighborhood are "density-connected" and assigned to the

same cluster. As density has to be comparable i.e. the density measure φ_S has to be unbiased with respect to the dimensionality of the subspace S (cf. Definition 2). This is achieved by normalizing with the expected density of the subspace dimensionality:

For any density measure φ_S with expectation $E[\varphi_S]$,

$$\frac{\varphi_S}{E[\varphi_S]} \text{ is dimensionality unbiased.}$$

For continuous attributes, our previous work on dimensionality unbiased subspace clustering provides such an unbiased density measure. Let $E_{cont}[\varphi_S]$ denote the expected density for a continuous valued subspace S. It is computed as the number of objects in the database DB multiplied by the volume ratio of the neighborhood in subspace S to the entire subspace S:

Definition 24. Continuous normalization

$$E_{cont}[\varphi_S] = |DB| \cdot \frac{vol(\varepsilon\text{-sphere}_S)}{vol(S)}$$

While this normalization by the expected density achieves an unbiased density, it still falls prey to the so called empty space problem as the expected density drops to almost zero for high dimensional subspaces. As already proposed for unbiased density-based clustering in Chapter 2, this effect can be tackled by introducing a minimum density (or frequency) to be fulfilled by the detected subspace clusters. For details, please refer to Section 2.3.

For heterogeneous data, computation of the expected density requires taking categorical attributes into account. By definition, categorical data attributes have no extension, i.e. only discrete values occur. As a consequence, distance values are discrete as well and the notion of ε-sphere neighborhoods leads to discontinuous densities.

We unify density assessment for categorical and continuous attributes. To ensure a consistent density measure, the expected density should be normalized for categorical attributes in the same manner as for continuous attributes. We achieve this consistency by treating categorical values not as discrete points, but as segments of the attribute value range. More precisely, the number of values v_i for each attribute dimension i of the categorical attributes is considered to be the value range extension in this attribute. The overall volume of a categorical subspace S_{cat} is then defined as the product of these ranges, yielding a rectangular overall volume $vol(S_{cat}) = \prod_{i \in Scat} v_i$. The expected density of categorical attributes is then the number of objects in the database DB multiplied by the ratio of the segment volume by the volume of the subspace.

Definition 25. Categorical normalization

$$E_{cat}[\varphi_S] = |DB| \cdot \frac{vol(segment)}{vol(S)}$$

with $vol(segment) = 1$ as each segment corresponds to one discrete value. For our running example the expected density for the categorical subspace $S_{cat} = \{d_5, d_6\}$ is simple computed by $18 \cdot \frac{1}{3 \cdot 3} = 2$ as we have 18 objects in our database and 3 possible values in each of the two categorical dimensions. This means that we expect to have two objects with the same categorical attribute value combination in our dataset. Thus, a meaningful 2d-categorical cluster in our example should have at least 2 objects, otherwise it has a frequency less than the expected. At least 6 objects form a meaningful 1d-categorical cluster.

This modified density computation corresponds to a frequency count in the categorical attributes, and fits in nicely with our continuous attribute normalization in the sense that the overall expected density normalization $E[\varphi_S]$ is consistent for both types of attributes in subspace $S = S_{cont} \cup S_{cat}$:

Definition 26. Heterogeneous normalization

$$E[\varphi_S] = |DB| \cdot \frac{vol(\varepsilon\text{-}sphere_{S_{cont}})}{vol(S_{cont})} \cdot \frac{vol(segment)}{vol(S_{cat})}$$

Thus, the overall normalization measures density of heterogeneous data objects by the degree of deviation from the expected density in continuous attributes and the expected frequency in categorical attributes.

7.4 Processing of heterogeneous data

Pattern detection in projections of high dimensional data is a computationally challenging task, especially in the presence of different data types. Using our consistent density measurement for both continuous and categorical attributes, one has to efficiently mine the heterogeneous subspace patterns. We propose an adaptive algorithm for efficient pattern detection in projections of heterogeneous data. In continuous data, density-based (subspace) clustering is known to be computationally complex, because it involves repeated data scans to identify the objects in the neighborhood [EKSX96]. This means that especially for subspace clustering, where the neighborhood has to be computed for different subspace projections, efficiency is a major concern. However, repeated database scans are not required for simple frequency computations on categorical data. Our algorithm for heterogeneous subspace clustering exploits such characteristics of continuous and categorical attributes in a novel index structure. This index is based on our previous work on efficient density-based subspace clustering for continuous values (cf. Chapter 6) and similar structures for frequent itemset mining [HPY00]. While these techniques are not applicable to heterogeneous data on their own, our index is adaptive to heterogeneous data and yields a significant efficiency improvement compared to recent subspace clustering approaches.

7.4.1 Continuous attributes

In the previous chapter, we have proposed an indexing structure, the SCY-tree for efficient mining of continuous valued subspace clusters in a depth-first fashion. As the SCY-tree has already been introduced in details, let us give only a short review to highlight the important parts for our novel index structure. The general idea is to index regions such that they can be evaluated for several different projections, and allow for merging of regions that might contain parts of a subspace cluster. Essentially, the SCY-tree approach benefits from the better efficiency of grid-based algorithms without losing the quality provided by density-based clustering approaches. The idea is to check not only each individual cell against a threshold, but to extend this in a density-based fashion as well: as long as there are objects within a neighborhood distance from the cell's border, we do not discard the cell, but instead merge it with this neighbor. Working in this fashion not only means efficient mining without loss of accuracy, but it also is the basis for our indexing structure that stores the compact cell counts, and also for just-in-time merging and mining of them.

This indexing structure, however, was devised only for continuous data types. Obviously, we could consider a naive solution of simply treating categorical attributes like continuous ones. However, as we will shortly see in greater detail, this would ignore the very properties of categorical data: merges of adjacent regions cannot occur, as in categorical values, this notion of range within a dimension does not exist. Simply "turning off" the merges would also be far from ideal, as we will see. We will analyze how we can reorganize the index such that heterogeneous data can be mined for subspace clusters in a far more efficient fashion.

7.4.2 Categorical attributes

As mentioned in Section 7.2.1 for categorical data the FP-tree mines frequent itemsets efficiently without candidate generation [HPY00]. The basic idea is to have all the needed information in this index structure to avoid expensive database scans. As in the SCY-tree after building the initial tree, mining is performed only on the tree structure. In Figure 7.3 we see the initial FP-tree for our running example. As the FP-tree can only handle categorical data we hide the first four continuous dimensions as one item "continuous". Proceeding recursively, the FP-growth algorithm restricts the initial FP-tree e.g. as depicted in the item "REPTILES" by extracting all paths ending in a node labeled with that item.

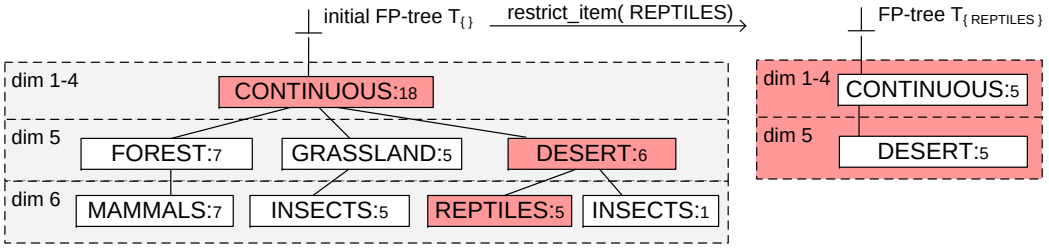


Figure 7.3: FP-tree example

Both the SCY-tree and the FP-tree from frequent itemset mining are related in the sense that they operate on compact node entries that represent cell counts. The SCY-tree, however, is capable of dealing with continuous values as well by applying merge operations on neighboring cells where necessary. However, such merge operations lead to computational overhead. Mining continuous data on the SCY-tree has higher runtime than mining on the FP-tree. The FP-tree, for both attribute types, on the other hand, is not able to handle continuous values.

7.5 Indexing heterogeneous attributes (HSM-tree)

As a straightforward solution to mining of heterogeneous data, we simply extend the ideas of SCY-tree and FP-tree (cf. reviews in Sec. 7.4.1 and Sec. 7.4.2) to heterogeneous data. Our novel HSM-tree structure can handle both categorical and continuous data types by automatically adapting to the actual data type. This adaptation makes the HSM-tree a non-trivial combination of two known tree structures. Furthermore, we introduce two novel strategies for adapting to different data types during tree construction.

Definition 27. HSM-tree structure
A HSM-tree T_{HD} represents a heterogeneous region $HD = \{hd_1, \ldots, hd_k\}$.
Each level in the HSM-tree represents one dimension. The order of dimensions $st(d_1) \ldots st(d_k)$ is given by (a subset of) the order $st(d_i) \leq st(d_j)\ \forall i < j$ with respect to strategy $st(d_i)$.
The HSM-tree consists of nodes, each of them stores:

- *a descriptor (d, i) for dimension d and*
  - *if d is continuous, interval i of the region and its count c of objects or -1 indicating a required merge*
  - *if d is categorical, item i and its frequency c*
- *a pointer to the parent node and a list of child pointers*
- *a pointer to a linked list of nodes with the same descriptor*

As one can easily see by comparing the above definition of the HSM-tree with the SCY-tree definition (Def. 22), we have two major differences: First, when building the HSM-tree, we differentiate between continuous and categorical attributes to compute the count for a region, or to record the frequency, respectively. Second, the HSM-tree follows a strategy (cf. Sec. 7.5.2) for building the tree. This strategy defines a favorable ordering of the dimensions for efficient mining. Please note that two major changes of HSM-tree are not explicit in the above definition: During mining on the HSM-tree, we use the consistent density normalization for heterogeneous data (cf. Def. 26), and, most importantly, we adapt to the heterogeneous attributes indicated by the descriptors in the HSM-tree (cf. following sections).

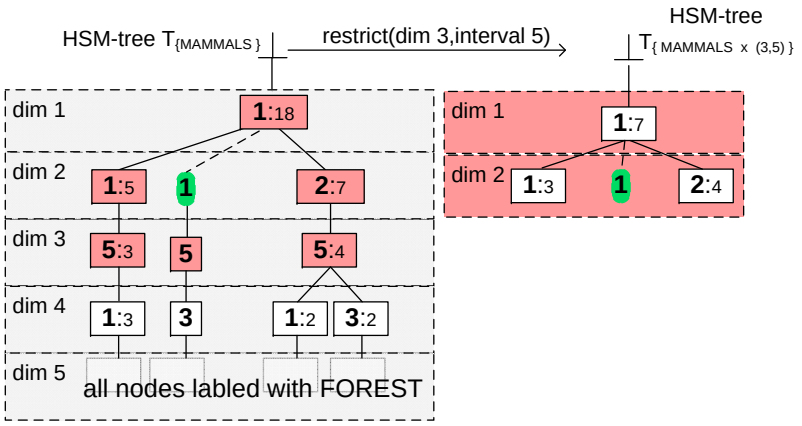


Figure 7.4: HSM-tree example

7.5.1 Adaptive mining

Based on the simple HSM-tree structure we perform a novel adaptive mining. Adaptation is required as there is a fundamental difference between the data types that merits more detailed consideration: There is no notion of ranges in categorical values. That is, only in continuous values a cluster might stretch across several grid cells in one dimension, because the values of the objects are neighboring, e.g. show temperature values that do not differ much. In categorical dimensions, this does not occur, e.g. either we are dealing with an insect or with a mammal, but in general there is no notion about "closeness" or distance between the two categorical values. This results in two effects for categorical values in HSM-trees: first, we do not need to check S-connectors, and second, we do not need to merge "cells" (i.e. attribute value combinations).

However, just by eliminating the corresponding nodes for S-connectors and omitting checks for merges, we are not doing as good as possible. We propose to reorganize the tree to achieve far better runtimes of the mining process on the HSM-tree. Basically, we observe that, the order of the dimensions in the HSM-tree matters with respect to runtime performance. Mining starts with the leaf nodes to extract the longest paths and thus also the high dimensional patterns. Thus having categorical values which are highly clustered at the leaf level would be beneficial for efficient mining.

Looking again at our running example we can construct the restricted HSM-tree for "MAMMALS" as shown in Figure 7.4. It also illustrates the advantage of combining both continuous and categorical information in a single tree structure. Having restricted the initial HSM-tree already to one categorical value, the highly correlated attributes vegetation and prevailing animals directly form a cluster. We only see "FOREST" nodes at the new leaf level, and all objects being represented by this HSM-tree also contribute to the hidden cluster. Furthermore, after the first restrictions in categorical values we obtain small tree structures. This is due to the fact that no merge operations have to be performed on these trees.

Adaptation to restrictions in both categorical and continuous attributes is the

key characteristic of adaptive mining using the HSM-tree. Thus, our novel tree structure enables the mining in heterogeneous subspaces of the data. Thus, in our running example, we can mine the heterogeneous subspace "animals" and "humidity" by restriction operations on the HSM-tree (cf. Figure 7.4). Performing another restriction on a different attribute type (e.g. in cell 5 in the continuous dimension 3) is quite easy as the whole mining process relies on a common basis. We thus easily can mine a heterogeneous pattern e.g. "MAMMALS" x (3,5) which represents the sensors surrounded by mammals and having a high humidity around 50%. The frequency/density of this attribute value combination can easily be determined based on the given HSM-tree. The sum of all nodels labeled with 5 in dimension 3 is equal to 7 (i.e. 7 objects support this pattern).

7.5.2 Adaptive tree construction

In construction of HSM trees, our goal is to ensure a structure that allows efficient mining of arbitrary patterns in heterogeneous attributes. According to Definition 27 the dimensions are ordered $d_1 \dots d_k$ with descriptors of d_1 at the root node and descriptors of d_k at the leafs of the HSM-tree. The order is defined by a sorting criterion $st(d_i)$ such that the ordered dimensions fulfill $st(d_i) \leq st(d_j)\ \forall i < j$. We study two different strategies for efficient mining. Both defining a different sorting of the dimensions in the HSM-tree.

Strategy 1 (ascending order): Reducing the tree size.
To reduce the size of the tree, one would sort dimensions with respect to the scattering of data in this dimension. A dimension in which the data is distributed over a wide range should not be inserted first in the tree as it would create a branching at the top of the tree. As an extreme, we would not want to first insert all continuous dimensions and then all categorical dimensions. This would clearly constitute a worst case scenario with respect to the resulting tree size. Instead, a dimension in which the data is clustered in one region leads to many common paths in the tree when inserted first. Many common paths reduce the size of the HSM-tree. In our first strategy, the idea is therefore to start by inserting the categorical dimensions to minimize tree size.

Strategy 2 (descending order): Reducing the number of merges.
As a drawback to the previous strategy, however, we observe that mining scattered dimensions with noisy data first forces many merges on large HSM-trees as no other dimensions have been restricted yet. These merges might not even lead to subspace clusters in the end as the $minSize$ threshold might only be exceeded due to the large size of the tree. Many merges can be avoided through a different processing order. By mining the scattered continuous dimensions last, we need to perform merges only on trees that have been restricted in several dimensions and are thus small. Merges on small HSM-trees are faster which is beneficial for the overall runtime. In this strategy, we focus on avoiding merges, and thus we insert the dimensions that contain clustered data or contain only categorical values last, i.e. in the opposite

order as in the first strategy.

To detect scatter, we use entropy as a measure for both strategies. Entropy is an information theoretic indicator for the homogeneity of the data and can be used to assess the distribution of objects in each dimension. Given the discretized data space for each dimension and thus also the percentage f_i of objects in each interval $i = 1 \ldots k$ entropy is calculated by $E(f_1, \ldots, f_k) = -\sum_{i=1}^{k} f_i \cdot \log(f_i)$. Inserting dimensions with low entropy values first (ascending order) realizes the first sorting strategy for small trees, while inserting dimensions with high entropy values (descending order) realizes the second strategy for fast subspace mining. Both ascending and descending order will be analyzed in experiments in the following section.

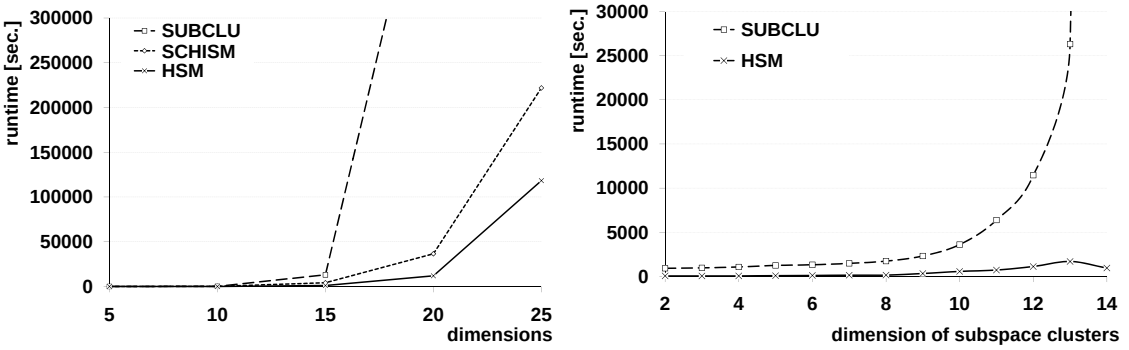
An overview over mining on HSM-trees is given in Algorithm 5. For any database, the algorithm uses only two database scans to build the HSM-tree. One scan for the computation of the dimension order with respect to the chosen strategy and a second scan to create the initial HSM-tree. To reduce the number of merge operations in later mining steps, we clearly differentiate between continuous and categorical attributes. In the mining phase, we proceed recursively on the dimensionality for any region that might contain subspace clusters. For any subspace region, we determine its unbiased density value according to the consistent definition for heterogeneous data given in Def. 26, and merge neighboring regions only where necessary (i.e. in the continuous attributes). The result of our algorithm is the set of all non-redundant heterogeneous subspace clusters, detected efficiently using the HSM-tree.

Algorithm 5 Heterogeneous Subspace Mining on HSM-trees

```
1: Algorithm HSM ( Database DB, Strategy st, set of dimensions D )
2: compute order d_1, ..., d_k, using strategy criterion st(d_i) {first scan of DB}
3: for each object in DB do
4:    for i=1 to k do
5:       if d_i continuous then
6:          insert grid cell and object count into initial HSM-tree
7:       else if d_i categorical then
8:          insert item and frequency into initial HSM-tree
9:       end if
10:   end for
11: end for
12: start recursive mining with initial HSM-tree T_{}
13: in each recursive step with given HSM-tree T_HD do:
14: for each descriptor (d, i) ∈ HD in the HSM-tree T_HD do
15:   restrict to T_{HD∪(d,i)}
16:   compute density in merged grid cells of T_{HD∪(d,i)}
17:   compute frequency for categorical items in T_{HD∪(d,i)}
18:   in each subspace, use heterogeneous normalization as in Def. 26
19:   recursive step for T_{HD∪(d,i)} until no further restriction possible
20: end for
21: output all detected heterogeneous subspace clusters
```

7.6 Experiments

We first show scalability of our heterogeneous subspace mining approach. As no other subspace mining algorithms exist on heterogeneous high dimensional data, we compare with two recent subspace clustering algorithms on continuous valued attributes. In a more detailed analysis we evaluate the impact of the heterogeneity of the data on the runtime of HSM. We show that our heterogeneous approach achieves efficiency improvement on categorical data by orders of magnitude. Due to the adaptive index structure we achieve improvements for the categorical parts of the data. On continuous attributes we can further improve efficiency by strategies for tree construction. For a thorough analysis, we evaluate runtime and memory performance of HSM with both ascending and descending sorting strategies.



(a) Data space (b) Hidden clusters

Figure 7.5: Scalability w.r.t dimensionality

Scalability w.r.t dimensionality. In our first experiment, we evaluate scalability with respect to the dimensionality of the subspace. For scalability experiments we generate synthetic data as in the previous chapters. We follow a method proposed in [KKK04] to generate density-based clusters in arbitrary subspaces. Figure 7.5(a) illustrates all three algorithms, HSM, SUBCLU and SCHISM, on datasets of dimensionalities 5 to 25. As we can see, SUBCLU, an apriori-based algorithm, deteriorates extremely at more than about 15 dimensions. SCHISM, a grid-based subspace clustering algorithm, performs much better than SUBCLU. Our HSM approach, however, clearly outperforms both competitors. HSM thus shows far better runtime performance than SCHISM, even though SCHISM is an approximative approach that does not mine the full result set.

Figure 7.5(b) illustrates the effect of varying subspace cluster dimensionalities. In a database of fixed dimensionality 15, subspace cluster dimensionality is varied from 2 to 15. As we can see, dimensionality of the subspace clusters is the decisive factor in runtime performance of SUBCLU. It shows reasonable runtimes up to about 10-dimensional subspace clusters. The breadth-first SUBCLU algorithm generates all lower dimensional projections, thus does not scale beyond this point. HSM is only

slightly affected by the dimensionality of subspace clusters as it avoids unnecessary generation of lower dimensional projections by its compact tree structure.

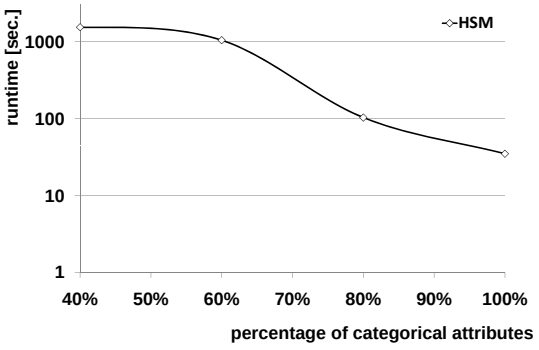
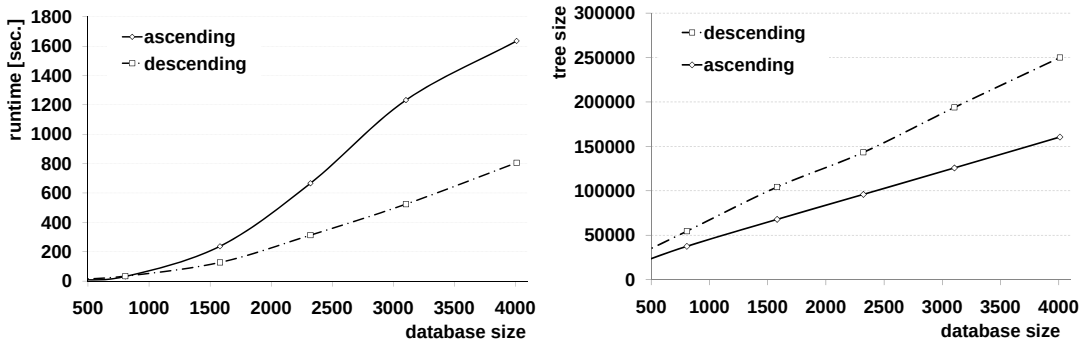


Figure 7.6: Variation of attribute types

Scalability w.r.t heterogeneity. We increase the percentage of categorical data to evaluate how the runtime of our HSM approach is affected by more and more categorical attributes. Due to the simpler frequency measurement in categorical data the algorithm should perform even better. In Figurer 7.6 we observe an efficiency improvement by orders of magnitude as we increase the percentage of categorical attributes. Obviously the HSM algorithm is able to adapt to the heterogeneous data. Being aware of the fact that for categorical data frequency is meaningful but also very efficient HSM achieves a significant runtime improvement for heterogeneous data.



(a) Runtimes (b) Tree size

Figure 7.7: Sorting strategies

Sorting strategies. We next evaluate the effect of different strategies for constructing the tree structure. As mentioned, the index construction mainly depends on the order of the dimensions. Two strategies have been proposed in the last section: creating compact trees using an ascending order of the dimensions w.r.t. their entropy or avoiding early merges by sorting the dimensions in descending order w.r.t. their entropy. We first evaluate the effect of the different sorting strategies on the

runtime (see Figure 7.7(a)). Clearly, avoiding early merges improves the overall runtime of HSM. Sorting in descending order almost halves the runtime. Thus, this sorting strategy is of major importance for improving the overall efficiency of our algorithm.

On the other hand, by sorting in ascending order the size of the initial tree is reduced as presented in Figure 7.7(b). However, the slight improvement in tree size has significant negative effect on the runtime. This shows that even with a smaller initial tree the mining algorithm cannot reach the good runtimes of the descending order heuristic. For efficient mining it is thus essential to have as few merges as possible in the first restrictions. The winning strategy is clearly observed in our experiments, as with slightly larger tree size one can improve the runtime significantly. This is achieved by the descending order heuristic which we recommend as a default sorting strategy.

7.7 Enabling subspace clustering on heterogeneous databases

In this work, we have discussed data mining in heterogeneous data. Most frequent itemset mining or (subspace) clustering algorithms focus on just one data type, continuous or categorical, respectively. We have presented a model that bridges the gap between continuous and categorical data types by providing a consistent view on interesting groups ob data objects. Our approach takes the expected density and the expected frequency of the subspace projections into account to uncover truly interesting patterns. Our algorithmic approach makes best use of indexing possibilities by identifying suitable ordering of the dimensions to achieve low runtimes. Our experiments demonstrate that our technique for heterogeneous data mining outperforms existing techniques. By adapting to the different characteristics of attribute types we further achieve a significant efficiency improvement for heterogeneous data.

Summing up, all of our first efficient algorithms (EDSC, INSCY and HSM) provide efficient and complete subspace clustering algorithms. Based on the DUSC model they provide efficient and complete subspace clustering computation. However, these two properties seem to be inapplicable for more complex subspace clustering models like in RESCU (cf. Chapter 3). As we have proven that our optimization of relevant subspace clusters is an NP-hard problem, we focus in the following chapters on approximative algorithms to achieve an efficient computation. We propose efficient algorithmic solutions which achieve high quality results even though they do not guarantee to provide the optimal clustering defined by our subspace clustering models. We focus on the development of a novel processing schema which uses our high quality density-estimation (cf. Chapter 8) and our steering heuristics (cf. Chapter 9) to achieve both efficient computation and high quality subspace clustering results.

Chapter 8

Density estimation for arbitrary subspace projections

We have compared subspace clustering and frequent itemset mining as major mining paradigms and unified both for pattern detection in continuous and categorical subspaces in Chapter 7. As common observation, both try to mine attribute combinations by utilizing processing schemes that include further attributes step-by-step. However, subspace clustering and frequent itemset mining via "step-by-step" algorithms that search the subspace/pattern lattice in a top-down or bottom-up fashion do not scale to large high dimensional databases. While in our previous work we have tried to enhance the efficiency of these traditional processing schemes, in the following chapters we propose to use an enhanced jump processing. The general idea of "jump" algorithms is to choose candidate subspace regions or patterns directly. However, their scalability and quality depend heavily on the rating of these candidates as mislead jumps incur poor results and costly candidate refinements. Existing techniques rely on simple statistics with low estimation quality or on inefficient database scans.

In this chapter, we propose $DensEst$, an efficient density estimator with significantly improved accuracy. It efficiently provides rough estimates of object counts in selective subspace regions. Furthermore, by incorporating correlations between dimensions, DensEst achieves not only efficient but also highly accurate estimations. We show how this density estimation technique can be easily integrated into subspace clustering and frequent itemset mining algorithms to improve both their efficiency and accuracy. Especially for subspace clustering, efficiency and accuracy of our novel density estimator are crucial for efficient and high quality algorithms. The DensEst approach forms a general filter step to reduce computation of subspace clusters to only the most promising subspace regions. Computing these subspace regions without costly database scans ensures scalability for large and high dimensional databases. We demonstrate the performance of our density estimation technique in thorough experiments and show its efficiency and accuracy improvement for existing algorithms.

8.1 Motivating density estimation techniques

Subspace clustering groups objects in individually relevant subspace projections. Similarly, frequent itemset mining identifies subsets of items that frequently co-occur in a set of transactions [AS94, HPY00]. Both mining tasks find dense/frequent attribute value combinations. However, as arbitrary projections are analyzed, these techniques have to search exponentially many candidates.

"Step-by-step" algorithms that search all combinations of patterns top-down or bottom-up do not scale to very high dimensional spaces [AGGR98, KKK04, AS94, HPY00]. As a consequence, "jump" algorithms have been proposed [KKRW05, ZYH+07]. The general idea is to quickly compute approximations of the subspace regions or itemset combinations that potentially contain dense or frequent patterns.

As an example, a sensor network in a forest surveillance project detects indicators of fire. These sensor nodes cluster together according to their measurements in a subset of sensors (temperature, humidity, dust-level). Typical measurements in a fire could be (high temperature, dry humidity, high dust-level). To detect if such an attribute value combination is a meaningful pattern, mining algorithms scan the data and compute the number of sensors that support this pattern (density of the pattern). The actual database access however is quite costly, which is especially disadvantageous if one has to perform this calculation for exponentially many attribute value combinations in subspace clustering or frequent itemset mining.

Density estimation therefore replaces the exact computation of density by an efficient rough estimate. This is especially interesting for jump algorithms, which need to efficiently estimate their jump destination in advance without accessing the data. For these algorithms, it is sufficient to get an efficient estimation of the actual number of sensors for the given attribute value combination. Efficiency is clearly required as the estimation procedure is used for exponentially many subsets of attributes the mining algorithm is interested in. Estimation has to be accurate as the algorithm has to rely on the estimation quality. Finally, one is interested in estimating a selective region in a subset of attributes. In our example for the chosen subset (temperature, humidity, dust-level), one was interested in the selective region (high, dry, high) but not in normal forest conditions with (low,wet,low) values. The aim is to achieve all three properties, namely efficiency, accuracy and selective estimation, for a good density estimation in high dimensional mining.

Existing jump algorithms, however, rely only on simple statistics for low dimensional projections to obtain interesting subspace regions in higher dimensional spaces. Such statistics lead to costly misleading jumps of poor pattern quality, e.g. simply using one-dimensional clusters with shared objects [KKRW05]. As no correlations between dimensions are considered, the choice may be off target and result in unnecessary expensive database scans for refinement. Likewise, pattern fusion of short itemsets requires repeated range queries for similar patterns to construct potentially large itemsets, incurring numerous database scans [ZYH+07]. In general, recent jump algorithms suffer from inaccurate density estimations, as they do not consider correlations between the attributes. These techniques are efficient, however, they perform misleading jumps and thus have low accuracy as they do not find the actual

patterns hidden in the data.

In this work, we propose a novel density estimation technique *DensEst* that achieves a better estimation combined with a much more efficient computation. By incorporating correlations between dimensions, DensEst shows far better accuracy. In addition, the better estimation quality is achieved by an efficient computation, as DensEst computes the accurate estimation by a new closed formula. DensEst is capable of computing the interestingness of selective subspace regions, based on compact local statistics that can be efficiently computed. DensEst can estimate density in arbitrary subspaces, showing improved estimation quality. As all the needed information is given in local statistics, the estimation process does not need any further database scans and thus is highly efficient. In addition, DensEst is capable of estimating a selective region without having to estimate all regions of a subspace, which is also advantageous for efficiency as it highly reduces the estimation computation costs.

DensEst thus fulfills the following key criteria for mining in large high dimensional databases:

- Accuracy - quality of the estimate is essential for the quality of the mining result in jump algorithms
- Subspace Efficiency - quick estimations of entire subspaces for scaling to high dimensional settings
- Region Efficiency - fast estimation of selected subspace regions is crucial for iterative jumping

8.2 Density estimation

For density estimation, first of all it is important to understand its application domain. In this section, we thus explain why data mining techniques benefit from accurate and efficient estimators. We therefore show drawbacks of recent approaches which are either not efficient or not accurate in their estimation. We generalize these algorithms to the concept of jump algorithms, before we explain our density estimation model. Based on this model, we provide a new closed formula for density estimation, which incorporates correlations between the attributes. Our DensEst approach can thus efficiently estimate density with a high estimation accuracy. Finally, we prove correctness of our estimator and provide algorithmic details for the efficient computation in DensEst.

8.2.1 Enhancing data mining algorithms

In the following, we illustrate how high dimensional data mining algorithms require accurate and efficient density estimates for high quality results and low runtimes. We describe two recent techniques from different data mining areas, starting with the

subspace clustering algorithm FIRES [KKRW05], before turning to Pattern Fusion for frequent itemset mining [ZYH+07].

We abstract from their details to give a general algorithm for approximative, scalable mining of patterns in high dimensional spaces based on low dimensional projections. In general, these mining techniques are based on a density calculation step, in which they determine whether a high dimensional candidate is promising. We improve these techniques by replacing this calculation with our efficient and highly accurate density estimation approach. It is crucial to develop an accurate estimator as the quality of the overall approximation is highly dependent on density estimation quality. Furthermore, efficient estimation is important to provide an overall scalable mining process.

Subspace Clustering. In high dimensional spaces, clusters are typically hidden by locally irrelevant attributes. Subspace clustering identifies clusters in relevant subspace projections of the full space. As the number of possible subspaces is exponential in the number of dimensions, this is a very complex process. Most subspace clustering algorithms use a bottom-up step-by-step algorithm to identify subspace clusters based on the apriori principle from frequent itemset mining [AS94]. Iteratively, clusters of dimensionality k are joined to generate candidates of dimensionality $k+1$, these candidates are clustered, and so on.

The drawback of such step-by-step methods is that all low dimensional projections have to be generated before high dimensional subspace clusters can be mined. This is a serious problem, as the very idea underlying step-by-step approaches is that high dimensional clusters are reflected in the low dimensional projections. Consequently, their number is generally huge, thus the algorithm has to work its way through tremendously large, low dimensional subspace clusters, before reaching high dimensional subspace clusters. High dimensional subspace clusters are typically considered to be the most interesting patterns, as they subsume their projections. Step-by-step subspace clustering is prohibitively slow in mining these interesting, or "maximal", subspace clusters in high dimensional data.

The idea in "jump" algorithms is to avoid the tedious clustering of huge low dimensional sets of clusters before reaching higher dimensional subspace clusters. FIRES starts by clustering all one-dimensional subspace projections $k = 1$ to identify *base clusters* [KKRW05]. The ensuing step is a direct jump to a high dimensional subspace cluster candidate $k \gg 2$ by combining several of the base clusters. This is based on the observation that any high dimensional subspace cluster will be reflected in many different one-dimensional base clusters.

For efficiency reasons the density computation is only performed in all one-dimensional spaces. This information is then used as a simple estimator for high dimensional projections. FIRES is thus an efficient approach that detects high dimensional subspace clusters. However, as only one-dimensional base clusters are used, the approximation quality is not always adequate for high dimensional spaces, resulting in many missed maximal subspace clusters. In the experimental section we will show that by replacing the simple one-dimensional estimator by our DensEst

approach, we improve both the overall subspace clustering quality and the runtime of FIRES.

Frequent Itemset Mining. The Pattern Fusion approach for frequent itemset patterns takes a similar jump approach [ZYH+07]. Starting from an initial pool of all frequent patterns up to a certain length, random seeds are drawn. Via range queries on these seeds, similar patterns are joined to a longer pattern. This process is repeated iteratively as long as more than K patterns are in the current pool. With Pattern Fusion, much longer patterns than with previous approaches are detected. However, for each Pattern Fusion step the actual frequency has to be calculated via a database scan. This exact frequency computation can be replaced by our density estimator, which improves the efficiency of the pattern fusion technique.

Generalized Jump Processing. For jump techniques in general, accurate and efficient estimation is crucial. If the estimate does not provide a close approximation of the true high dimensional pattern, that pattern may not be detected. Likewise, the efficiency of these methods largely depends on the efficiency of the estimate computation. In the following, we give a generalized view of such jump algorithms and explain how we can improve them both in terms of accuracy and efficiency by our proposed density estimation method DensEst.

We abstract from these two algorithms to a general view, as sketched in Algorithm 6. Both algorithms start from an initial set of patterns, from which they combine new candidates. For the combined patterns, an estimate of the density is computed. This estimation is either performed by exact density calculation via a database scan or a rough estimation. In the last step the combined pattern is refined, provided that the density estimate indicates a promising candidate. Optionally, the process can be repeated based on this novel pool of patterns.

Algorithm 6 Generalized jump processing

```
1: input: initialPool
2: candidateSet = combineCandidates(initialPool);
3: for each candidate ∈ candidateSet do
4:    quality = densityEstimation(candidate);
5:    if quality > threshold then
6:       isPattern = refine(candidate);
7:       if isPattern then
8:          return refined candidate;
9:       end if
10:   end if
11: end for
```

From this general algorithm, we can easily see that the costly refinement step is carried out whenever the density estimate suggests to do so. Consequently, the accuracy of the density estimate is crucial for the quality of any jump algorithm.

As the density estimator is called for any potential candidate, its efficiency is also decisive for the overall runtime. As the estimation is the central step in this generalized processing, we will first give the basic notions and explain recent estimation approaches in Section 8.2.2, before we propose our DensEst approach in Section 8.2.3. In contrast to existing methods, our novel DensEst approach provides both efficient and accurate estimates for good and fast mining results. DensEst achieves accuracy by incorporating correlations between the attributes. While taking all correlations into account typically leads to high runtimes during the computation of the estimate, we have developed several heuristics for an efficient but still accurate estimate computation (cf. Section 8.2.4).

8.2.2 Modeling density estimation

Before jumping to a high dimensional projection the estimation technique ensures that there is a high probability for meaningful patterns in this region. Assuming categorical attributes, a region can be described by a set of interesting dimensions/attributes S and the attribute values j_i for each attribute $i \in S$ (w.l.o.g. $S = \{1, \ldots, s\}$). As a consequence, an estimator is a function f_S that calculates a value $f_S(j_1, \ldots, j_s)$ as an approximation of the true number of objects for the specified region $(j_1, \ldots, j_s)$ in subspace S. For real-valued attributes this setting is achieved by a discretization of the data space. In both cases the function f_S yields a $|S|$-dimensional histogram which estimates the number of objects in the selected subspace S.

In a toy example with two intervals per dimension (Fig. 8.1) we have 14 objects in a 3d space. As most (10) of the objects are located in region $(2, 1, 1)$ while only two objects are in $(1, 1, 1)$ and single objects are in regions $(1, 2, 2)$ and $(2, 2, 1)$, an accurate estimator should calculate that it is highly probable to find objects in $(2, 1, 1)$. Hence, the estimator $f_{\{1,2,3\}}(2, 1, 1)$ should give this region the highest value for subspace $\{1, 2, 3\}$.

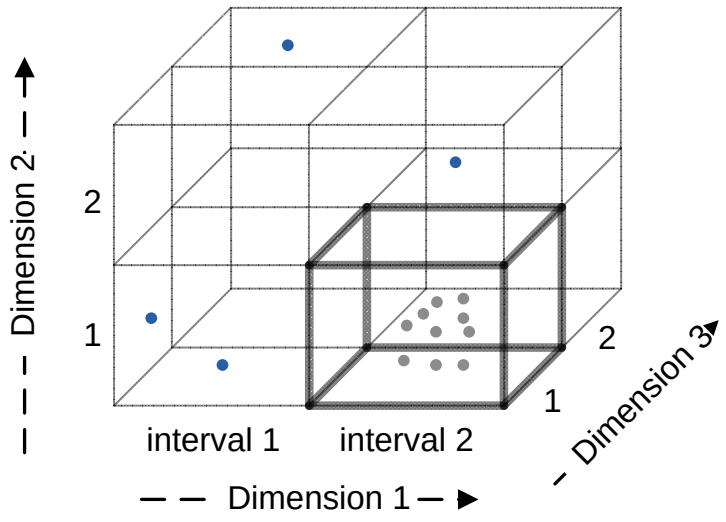


Figure 8.1: Data space example in 3d

We assume w.l.o.g. that the estimation function is normalized, such that the overall sum of the histogram bins is equal to 1. Thus the function f_S can be interpreted as the probability that an object is located in one of the discretized regions. The density estimation histogram represented by f_S can thus also be interpreted as the probability that the random variables X_i (representing dimension i) take the values j_i:

$$p(X_1 = j_1, \ldots, X_s = j_s) = f_S(j_1, \ldots, j_s)$$

We use both interpretations in this work for different goals: The estimation function is used for the description of different estimators and the components they are based on. To derive the density estimator formula and to prove its correctness, we use probabilities.

The general idea to obtain f_S is using information on the low dimensional projections of the data space.

1D estimation. A simple approach to determine such an estimation is by considering the one-dimensional (marginal) histograms [PHIS96]. Naively assuming independence between all the dimensions one can compute the estimation in a high dimensional region by combining the given histograms:

$$f_S^{1D}(j_1, \ldots, j_s) := \prod_{i \in S} f_{\{i\}}(j_i)$$

The estimation is simply the product of the histogram values in the specified one-dimensional intervals j_i. Obviously, this can be done efficiently as after having created the one-dimensional histograms once, the product can be computed for arbitrary regions and subspaces S. However, the assumption of independence is also obviously not true in general. This assumption leads to qualitatively bad estimations as the correlations between dimensions are not considered. Especially the relevant dimensions of a subspace cluster show high correlations, which is a key characteristic of such a subspace cluster.

In summary, the 1D histogram technique uses insufficient information for the estimation. Consequently, mining algorithms based on such estimates like FIRES suffer from poor estimation quality and thus almost random jumps into high dimensional spaces [KKRW05].

xD estimation (IPF technique). Although 1D techniques are efficient solutions for estimating density, they provide only a poor estimation quality. A direct extension of such 1D approaches are xD techniques, which use higher dimensional ($x > 1$) histograms as basis for the estimation. As such techniques use more information it is clear that one can thus increase estimation quality. The two main advantages are that by the given 2d, 3d or higher dimensional histograms the data distribution is described by more histogram bins. The second and even more important property is that the xD histograms contain dependencies between dimensions. Correlations can thus be incorporated into the estimation process and achieve a significant quality

improvement. The problem with such xD histograms is that a direct computation of density in arbitrary subspaces is not possible [BFH95]. An approximate computation is the *Iterative Proportional Fitting (IPF)*, that has already been applied for selectivity estimation and also approximative query processing [BFH95, MMK+05, PK01]. The major drawback of the IPF approach is, that every region in a subspace has to be computed. Assuming 10 intervals per dimension one has to compute and store $10^{|S|}$ estimations in multiple iterations. This results in high estimation runtimes even for medium dimensional subspaces S. Although IPF provides better estimation, it takes far too much time.

8.2.3 Density estimation (DensEst)

As discussed in the previous section, one dimensional histogram estimators typically do not provide the necessary accuracy required for high dimensional spaces. The general idea in our density estimation approach is therefore to decompose the density within a high dimensional region into a product of its two-dimensional projections, for which we computed the 2d histograms. By using 2d histograms we are able to model correlations between dimensions.

We avoid the drawbacks of IPF by giving a direct computation method to achieve an efficient estimation process. The key idea is to relax some correlations between dimensions to keep the property of direct computation as observed for 1D histograms. Consequently, our DensEst technique needs only 2d histograms to estimate the density, and it does not require iterative processing of all s-dimensional histogram bins as in IPF, since we provide a closed formula. Especially in the case of selective estimation, as given for jump algorithms, DensEst provides an efficient estimation. By having a closed formula at hand, DensEst achieves an efficient estimate of one s-dimensional histogram bin without the need of computing estimates for all other bins in subspace S.

For the decision which are the relevant correlations between dimensions we first model the correlations in independence graphs. In a second step we propose our new direct estimator out of 2d histograms and present our strategies to choose the relevant correlations.

Graph representations. To represent correlations between dimensions, we use independence graphs [Whi90]. Each node represents a random variable, i.e. a dimension in the subspace S, edges denote dependencies. For density estimation, we denote the probability of an object to be located within a specific region j_i in dimension i as $p(X_i = j_i)$. As an abbreviation we use $X_A = \{X_i | i \in A\}$ for a set of random variables.

The random variables X_i and X_j are conditionally independent given all other variables $X_{V\setminus\{i,j\}}$ (short: $X_{\{i\}} \perp X_{\{j\}} | X_{V\setminus\{i,j\}}$), iff there is **no** edge between node i and j. An example is given in Figure 8.2. Each of the six nodes represents a random variable/dimension. For example, $X_{\{4\}}$ and $X_{\{5\}}$ are conditional independent given $X_{\{1,2,3,6\}}$, i.e. $X_{\{4\}} \perp X_{\{5\}} | X_{\{1,2,3,6\}}$, since there is no connecting edge between them.

Formally,

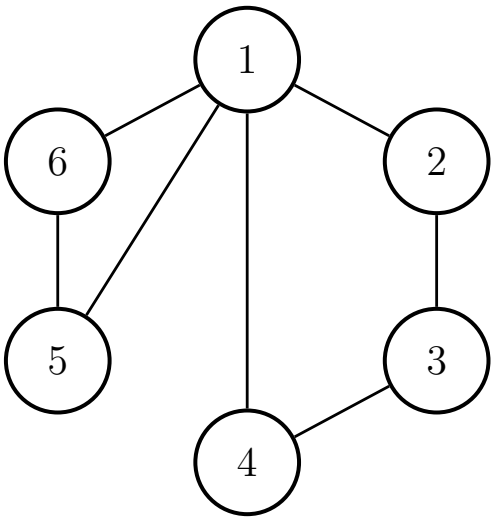


Figure 8.2: Independence graph example representing the correlations between dimensions in a 6d subspace

Definition 28. Independence graph.

An independence graph is an undirected graph $G = (V, E)$ *of vertices* $V = \{1, \ldots, n\}$ *and edges* E *with:*

$$(i, j) \notin E \text{ iff } X_{\{i\}} \perp X_{\{j\}} | X_{V \setminus \{i,j\}}$$

A key property for our approach is that for acyclic independence graphs we can derive a direct estimator. In the following we derive such a direct estimator assuming an acyclic independence graph. We then present our strategies to obtain the necessary acyclic graph.

Density estimation is based on the dependencies between the probability of objects to be located within a certain region. Based on the locality information in low dimensional spaces, we estimate the locality in higher dimensional projections, using the conditional probabilities. In the independence graph representation this means that exactly those regions form the basis of density estimation that are connected via edges in the independence graph.

For any connected pairs, i.e. $(k, l) \in E$, assume the corresponding 2d histogram denoted as $f_{\{k,l\}}$ is given. Then, the density of any high dimensional region j in a subspace S, specified through its set of dimensions $\{1, \ldots, s\}$, can be computed as the product of all co-dependent 2d histograms. If any node is included more than once, i.e. its joint probability with several nodes is part of the computation, we have to normalize the product by dividing with its individual probability.

Given the independence graph in Figure 8.3 and our previous toy example (cf. Fig. 8.1), we want to estimate the density of the region at the lower front bottom. Therefore we compute the 2d histograms $f_{\{1,2\}}$ and $f_{\{1,3\}}$ that correspond to the two edges $(1, 2)$ and $(1, 3)$ in the independence graph. At the same time we infer for each node i the 1d histogram $f_{\{i\}}$. The link between the independence graph and the histograms is illustrated in Figure 8.3.

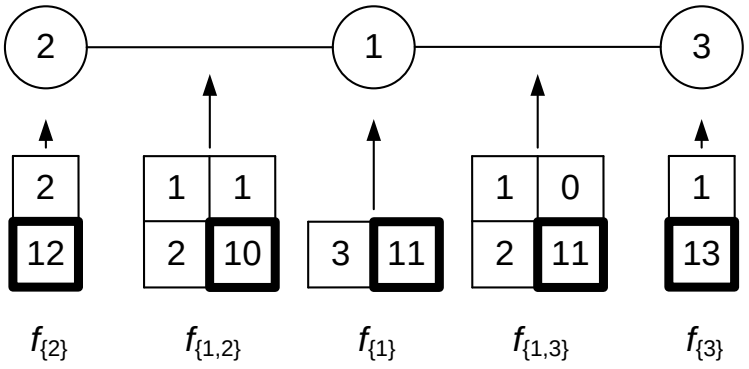


Figure 8.3: Independence graph & histograms

Theorem 8. Density estimator.
Let $G = (V, E)$ undirected acyclic connected independence graph with $V = \{1, \ldots, s\}$. The density of a cell j in subspace $S = V$, $f_S(j_1, \ldots, j_s)$, is computed as

$$f_S(j_1, \ldots, j_s) = \frac{\prod_{(k,l)\in E} f_{\{k,l\}}(j_k, j_l)}{\prod_{m\in V} \left[f_{\{m\}}(j_m)\right]^{deg(m)-1}} \tag{8.1}$$

where $deg(m)$ denotes the degree of node $m \in V$.

Note that we assume positive histogram values. If an entry is zero, the estimate is obviously zero as well. The 1d histograms $f_{\{m\}}$ for all $m \in V$ are implicitly given, since G is a connected graph.

Thus the density $f_S(j_1, \ldots, j_s)$ of a region $j = (j_1, \ldots, j_s)$ in subspace S is computed as the product over all connected histograms $f_{\{k,l\}}(j_k, j_l)$ in dimensions $\{k, l\}$. If any $f_{\{m\}}(j_m)$ occurs more than once, i.e. the corresponding node has degree larger than one, we have to account for this by normalizing accordingly.

For the example in Figures 8.1 and 8.3, the density of region (2,1,1) in subspace $\{1, 2, 3\}$ is computed via

$$f_{\{1,2,3\}}(2,1,1) = \frac{f_{\{1,2\}}(2,1) \cdot f_{\{1,3\}}(2,1)}{f_{\{1\}}(2)^{2-1} \cdot f_{\{2\}}(1)^{1-1} \cdot f_{\{3\}}(1)^{1-1}} = \frac{10 \cdot 11}{11^1 \cdot 12^0 \cdot 13^0} = 10$$

We prove correctness of our density estimator from Theorem 8. The basic idea is straightforward and relies on decomposition of the joint probability into conditionally independent subgraphs, using the fact that the graph is acyclic.

We first note the following lemma:

Lemma 1. Conditional independence.
For $G = (V, E)$ connected acyclic independence graph with at least two nodes, $l \in V$ leaf, $k \in V$ parent of l, $R = V \backslash \{l, k\}$ the remaining nodes, we have:

$$\forall R' \subseteq R: \; X_{\{l\}} \perp X_{R'} | X_{\{k\}\cup(R\backslash R')}$$

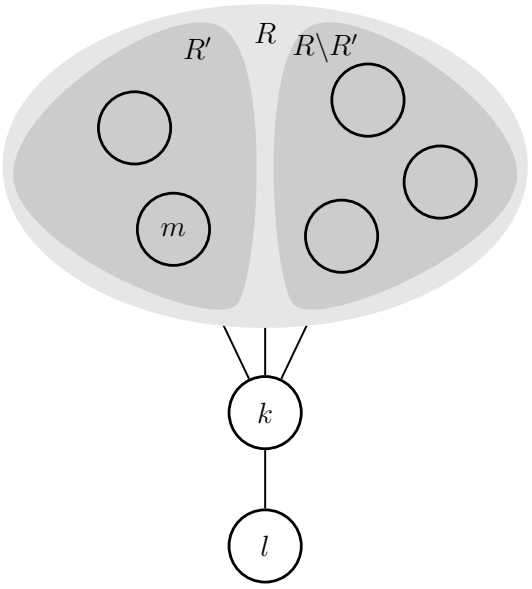


Figure 8.4: Independence graph

The leaf X_l is thus independent of the variables in any subset of the remaining nodes R', given the parent X_k and the other variables in $R\backslash R'$ as illustrated in Figure 8.4.

Proof. We use induction over the cardinality n of R'.

Induction basis: $n = 0 = |R'| \Rightarrow R' = \{\}$

$$\begin{aligned} p(X_l, X_{\{\}}|X_{\{k\}\cup R\backslash\{\}}) &= p(X_l|X_{\{k\}\cup R\backslash\{\}}) \\ &= p(X_l|X_{\{k\}\cup R\backslash\{\}}) \cdot p(X_{\{\}}|X_{\{k\}\cup R\backslash\{\}}) \\ &\Leftrightarrow X_{\{l\}} \perp X_{R'}|X_{\{k\}\cup(R\backslash R')} \end{aligned}$$

Induction hypothesis: Assume Lemma 1 for all R', $|R'| \leq n$.

Induction inference: $n \mapsto n+1$

Let $|R'| = n+1$, $m \in R'$, $R'' = R'\backslash\{m\}$.

Since l is a leaf, and $m \neq k$ is not the parent of l, there is no edge between m and l, and independence holds:

$$X_{\{l\}} \perp X_{\{m\}}|X_{V\backslash\{l,m\}} \Leftrightarrow X_{\{l\}} \perp X_{\{m\}}|X_{\{k\}\cup(R\backslash R')\cup R''}$$

Since R'' has cardinality n, the induction hypothesis yields:

$$X_{\{l\}} \perp X_{R''}|X_{\{k\}\cup(R\backslash R'')} \Leftrightarrow X_{\{l\}} \perp X_{R''}|X_{\{k\}\cup(R\backslash R')\cup\{m\}}$$

We use the intersection rule of conditional independence [Pea00] $X_A \perp X_B|X_{C\cup D} \wedge X_A \perp X_C|X_{B\cup D} \Rightarrow X_A \perp X_{B\cup C}|X_D$ with $A = \{l\}$, $B = \{m\}$, $C = R''$, and $D = \{k\} \cup (R\backslash R')$:

$$X_{\{l\}} \perp X_{\{m\}\cup R''}|X_{\{k\}\cup(R\backslash R')} \Leftrightarrow X_{\{l\}} \perp X_{R'}|X_{\{k\}\cup(R\backslash R')}$$

□

We are now ready to prove Theorem 8 using the above decomposition of the graph to show that the joint probability computed in our density estimator can be decomposed successively as well.

Proof. via induction over the cardinality of the independence graph ($|V| = n$).

Induction basis: $n = 1 = |V| = |S|$

The graph V has a single node m of degree zero:

$$p(X_S) = p(X_{\{m\}}) = \frac{1}{\left[p(X_{\{m\}})\right]^{-1}} = \frac{1}{\left[p(X_{\{m\}})\right]^{deg(m)-1}}$$

Induction hypothesis: Assume Theorem 8 for all connected acyclic independence graphs with $|V| \leq n$.

Induction inference: $n \mapsto n+1$

Let $G = (V, E)$ connected acyclic independence graph with $|V| = n+1$. Since G is acyclic, there is a leaf $l \in V$ with parent $k \in V$:

$$\begin{aligned}
& p(X_{\{l\}}, X_{V\setminus\{l,k\}}|X_{\{k\}}) \overset{\text{lemma 1}}{\underset{[R'=V\setminus\{l,k\}]}{=}} \\
& p(X_{\{l\}}|X_{\{k\}}) \cdot p(X_{V\setminus\{l,k\}}|X_{\{k\}}) \\
\overset{p(X_{\{k\}})>0}{\Leftrightarrow} \quad & p(X_{\{l\}}, X_{V\setminus\{l,k\}}|X_{\{k\}}) \cdot p(X_{\{k\}}) = \\
& p(X_{\{l\}}|X_{\{k\}}) \cdot p(X_{V\setminus\{l,k\}}|X_{\{k\}}) \cdot p(X_{\{k\}}) \\
\Leftrightarrow \quad & p(X_{\{l\}}, X_{V\setminus\{l,k\}}, X_{\{k\}}) = \\
& p(X_{\{l\}}|X_{\{k\}}) \cdot p(X_{V\setminus\{l,k\}}, X_{\{k\}}) \\
\Leftrightarrow \quad & p(X_V) = p(X_{\{l\}}|X_{\{k\}}) \cdot p(X_{V\setminus\{l\}}) \\
\Leftrightarrow \quad & p(X_V) = \frac{p(X_{\{l,k\}})}{p(X_{\{k\}})} \cdot p(X_{V\setminus\{l\}}) \qquad (8.2)
\end{aligned}$$

We decompose leaf l and the remaining graph to obtain a smaller G': $V' = V\setminus\{l\}$, $E'=E\setminus\{(l,k)\}$, $G' = (V', E')$ with $p(X_{V'}) = p(X_{V\setminus\{l\}})$. G' is also connected and acyclic with cardinality n. The degree of nodes in G' is decreased by one for the parent of l, since their connecting edge is removed: $deg'(m) = deg(m) \forall m \neq k$ and $deg'(k) = deg(k) - 1$. Using the induction hypothesis, we obtain:

$$p(X_{V\setminus\{l\}}) = p(X_{V'}) = \frac{\prod\limits_{(k',l')\in E'} p(X_{\{k',l'\}})}{\prod\limits_{m'\in V'} \left[p(X_{\{m'\}})\right]^{deg'(m')-1}} \qquad (8.3)$$

We have:

$$p(X_V) \overset{(8.2)}{=} \frac{p(X_{\{l,k\}})}{p(X_{\{k\}})} \cdot p(X_{V\setminus\{l\}}) \overset{(8.3)}{=}$$

$$\frac{p(X_{\{l,k\}})}{p(X_{\{k\}})} \cdot \frac{\prod\limits_{(k',l')\in E'} p(X_{\{k',l'\}})}{\prod\limits_{m'\in V'} \left[p(X_{\{m'\}})\right]^{deg'(m')-1}}$$

$$= \frac{\prod_{(k,l)\in E} p(X_{\{k,l\}})}{\prod_{m\in V} \left[p(X_{\{m\}})\right]^{deg(m)-1}}$$

□

Thus, the density estimation of any region $(j_1, \ldots, j_s)$ in subspace S, which corresponds to the probability $p(X_V)$ with $V = S$, can be computed as stated in Theorem 8 via the 2d histograms and the independence information.

8.2.4 Graph construction strategies

So far, we have assumed that an acyclic independence graph is given. For density estimation of any subspace region, we have to construct such an acyclic graph by choosing some relevant edges (cf. Figure 8.5). Obviously, the goal is to construct the acyclic graph such that the density estimation approximates the "true" distribution as closely as possible, and that the computation is efficient. We present two different strategies for the construction of independence graphs.

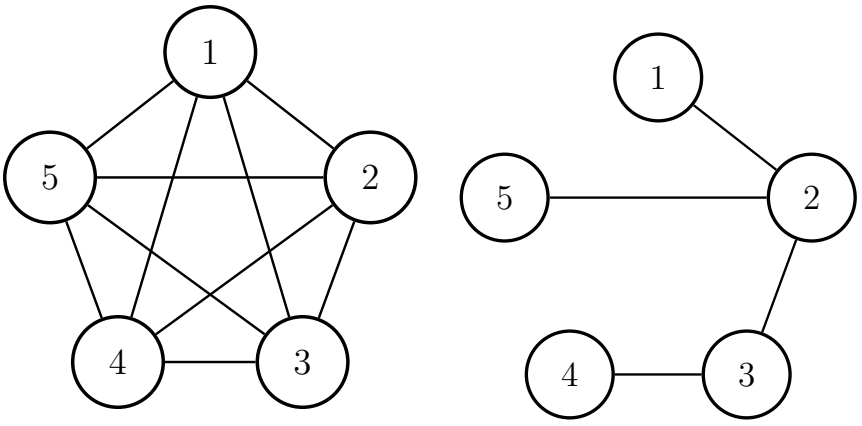


Figure 8.5: Construction of acyclic independence graph

2d-χ^2 strategy. Constructing an acyclic graph from the complete independence graph corresponds to a local independence assumption. As edges are removed, we ignore conditional probabilities. Choosing edges for removal should therefore focus on edges where the independence assumption is close to the actual relationship between the vertices. In statistics, a measure for the validity of independence assumption is the χ^2 test.

For a 2d histogram $f_{\{k,l\}}$ we have:

$$\chi^2_{kl} = \sum_{\substack{\text{all cells } (a,b) \\ \text{of histogram } f_{\{k,l\}}}} \frac{\left[f_{\{k,l\}}(a,b) - E_{\{k,l\}}(a,b)\right]^2}{E_{\{k,l\}}(a,b)}$$

where $E_{\{k,l\}}(a, b)$ denotes the expected density value of any cell assuming independence of dimensions, i.e.

$$E_{\{k,l\}}(a, b) = f_{\{k\}}(a) \cdot f_{\{l\}}(b)$$

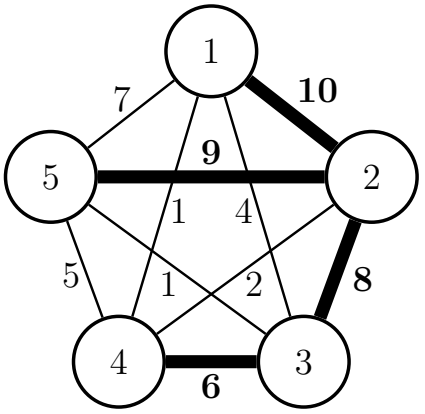


Figure 8.6: Complete graph with χ^2 values. Bold lines indicate the maximal spanning tree.

Figure 8.6 illustrates the χ^2 values for 2d histograms. An edge (k, l) corresponds to a histogram $f_{\{k,l\}}$. Low χ^2 values are interpreted as "more independence".

Our statistical strategy thus chooses histograms with large χ^2 values to keep those edges that indicate most dependences. To obtain the best possible spanning tree from the graph, i.e. a fully connected graph with no cycles, with respect to the χ^2 independence test, we choose those edges whose weights maximize the sum of χ^2 values. In the example in Figure 8.6, we thus choose to keep the edges in bold lines to connect all vertices without introducing cycles.

Randomized strategies. The χ^2 strategy is based on an evaluation of all 2d histograms, which have to be computed before the spanning tree is constructed. This is also the case, even if only one region has to be estimated.

For the example in Figures 8.1 and 8.3 the density of the region $f_{\{1,2\}}(1, 2) = 1$ has to be computed for the χ^2 value, but this density is not used for the estimation of $f_{\{1,2,3\}}(2, 1, 1)$ at all. To avoid this overhead, we propose an alternative randomized strategy. The idea is to simply choose edges in the graph randomly such that all vertices are connected without cycles. We propose three heuristics, as illustrated in Figure 8.7: a random tree, random path, or random star on the complete graph.

They differ mainly in the variance of node degrees. In a path structure, almost all nodes have the same degree, whereas the star center node has a much higher degree than the outer nodes. The random tree is in-between these two extremes.

The importance of the node degrees is reflected in the density estimator formula (cf. Theorem 8), more precisely in the denominator:

$$\prod_{m \in V} \left[f_{\{m\}}(j_m) \right]^{deg(m)-1}$$

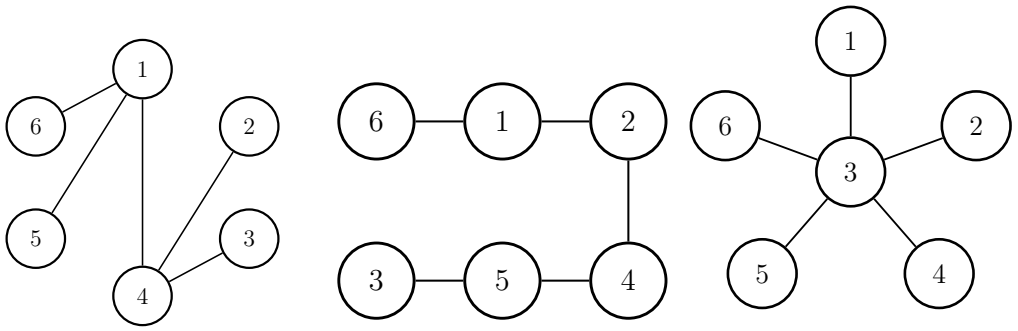


Figure 8.7: Randomized strategies

For the three heuristics in Figure 8.7 the denominator yields:

$$\begin{aligned}
\text{tree:} \quad & [f_{\{1\}}(j_1)]^2 \cdot [f_{\{4\}}(j_4)]^2 \\
\text{path:} \quad & [f_{\{1\}}(j_1)]^1 \cdot [f_{\{2\}}(j_2)]^1 \cdot [f_{\{4\}}(j_4)]^1 \cdot [f_{\{5\}}(j_5)]^1 \\
\text{star:} \quad & [f_{\{3\}}(j_3)]^4
\end{aligned}$$

This means, that in the star heuristics, only the 1d histogram in dimension 3 is used which consequently has a strong influence on the resulting density estimate. As a consequence the estimation is very sensitive to the choice of the center node and fluctuating results could arise. To avoid this effect and obtain greater robustness to the choice of nodes and edges the path strategy is presented. As one can see, almost all 1d histograms are taken into account, and no dimension is favored over the other. However in this way uncorrelated dimensions could impair the density estimation. The tree method lies in between these extrema and is less sensitive than the star, yet less robust than the path heuristics.

These three heuristics allow very efficient density estimates for regions, even without complete computation of all histograms. By the use of 2d histograms all approaches nevertheless take correlations among the dimensions into account. We evaluate these strategies in the experiments.

8.3 Experiments

We evaluate the efficiency and accuracy of our DensEst approach on several real world data sets from the UCI Machine Learning Repository [AN07], the UCR Time Series Classification/Clustering Page [KXWR06] and the Frequent Itemset Mining Dataset Repository [Fre08].

We first evaluate the density estimator by comparing it to the IPF and 1d histograms estimators discussed in Section 8.2.2, and then turn to experiments on the estimator integrated into data mining algorithms.

We report runtime and quality improvement factors for the different data sets to illustrate the speed-up and quality gain. We also report the variance in the deviation

from the exact calculation, since the absolute difference is not as important as the consistency of the estimate. Consistent estimation means that the densest region is indeed rated as the most promising candidate, which is the crucial decision made in jump data mining algorithms.

We integrated our density estimator into two jump data mining algorithms, FIRES for subspace clustering, and Pattern Fusion for frequent itemset mining [KKRW05, ZYH+07]. The density estimator provides an efficient way of determining the likely quality of a candidate before running the costly clustering or frequency counting procedures. Similarly, our density estimation can be used in any other data mining algorithm as an approximation of costly scans for e.g. density or frequency computations.

All implementations are in Java, and runtime experiments were run on Pentium 4 computers with 2.4 GHz and 1 GB main memory.

Table 8.1 summarizes the datasets used for the subspace clustering algorithm. To measure the accuracy on real world data where no ground truth on the number and size of subspace clusters is known, we use class label information in the data as ground truth. For the itemset mining algorithm we make use of the three datasets Connect, Mushroom and Pumsb [Fre08]. Additionally the synthetic dataset Diag30, presented in [ZYH+07], was used. Unless stated otherwise, we use the Pendigits dataset with 16 dimensions as default data set for the following experiments.

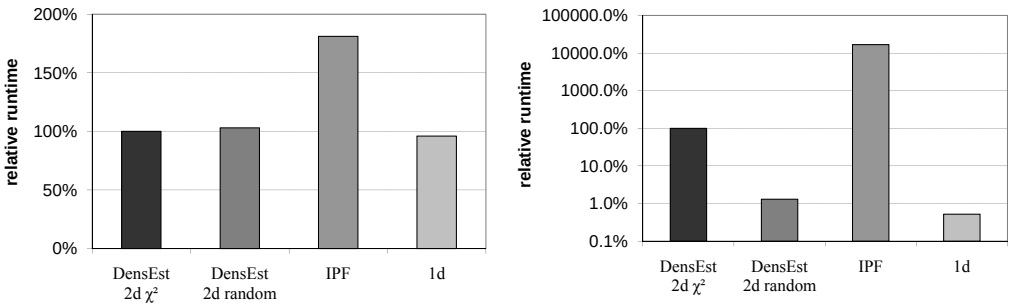
	Objects	Dimensions	Classes	Source
Pendigits	7494	16/32	10	[AN07]
Signs	700	20	10	[AN07]
Leaf	1125	25	15	[KXWR06]
Face	2250	25	14	[KXWR06]

Table 8.1: Data sets - Subspace Clustering

8.3.1 Density estimators efficiency

We evaluate the efficiency of our DensEst technique in comparison with competing techniques IPF and 1d histograms. In Figure 8.8(a) and 8.8(b) the 2d tree strategy is used as a representative for the randomized strategies. The resulting runtimes of the two other approaches (2d star, 2d path) are nearly identical.

As we can see in Figure 8.8(a), estimation of density for entire subspaces is far more efficient using DensEst. The IPF technique takes almost twice as long to compute an estimate, whereas the simpler 1d histograms are only slightly faster. As we can see, there is hardly any difference between the χ^2 graph construction strategy and the randomized strategies. The overhead in the χ^2 method, i.e. the calculation of the χ^2 values for all 2d histograms, is negligible in comparison to the computation effort for the density estimation of all regions in the subspace. Thus the runtime difference is only marginal for all DensEst strategies.



(a) subspace estimation (b) region estimation

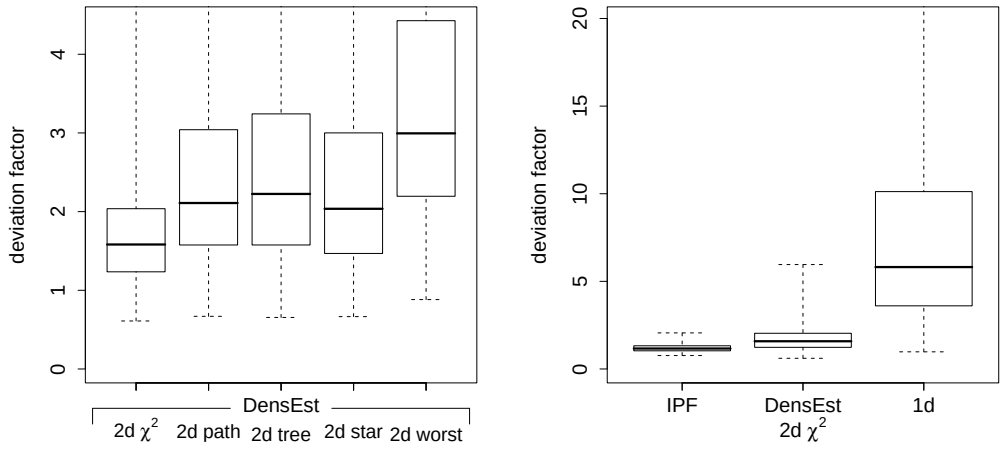
Figure 8.8: Efficiency of density estimation

Our next experiment evaluates the runtime for estimating selective regions in a subspace. As we can see, the differences in runtimes are far greater (note that the runtime is on a logarithmic scale). The simple 1d histograms are by far the fastest approach. Our DensEst technique performs very well, as it is capable of computing selective 2d histograms for estimation, as can be seen in the randomized strategy runtime. The χ^2 strategy requires far more overhead to estimate single regions, because the calculation of the χ^2 values is still based on the full 2d histograms, but for the estimation itself only a fraction of the histogram is required. Thus the method takes almost two orders of magnitude longer. IPF, however, is clearly not suitable for estimation of individual regions. It has to compute the entire subspace estimates in an iterative process. Consequently, its runtime is more than two orders of magnitude higher than that of the χ^2 strategy, and even about four orders of magnitude higher than the randomized strategies.

8.3.2 Density estimators accuracy

We evaluate the effect of construction strategies for independence graphs in the DensEst technique on the accuracy of the estimate. We compare the χ^2 strategy to the three heuristics of randomized independence graph construction, the path, tree, or star (cf. Section 8.2.4). As a baseline comparison, we include the "opposite" of the χ^2 strategy, termed 2d worst, i.e. based on the χ^2 analysis, we always remove the edge with the greatest dependence. This should result in a worse estimation. In Figure 8.9(a), we illustrate the resulting variance in deviation from the exact computation. As we can see, the χ^2 strategy is indeed the best strategy in terms of accuracy. It is extremely robust in the variance from the exact value, and thus provides consistent estimates for data mining. The randomized strategies show greater variance, but still are substantially better than the 2d worst approach. For illustrative purposes, we cut off the diagram, but the maximal deviation factors correspond to the quantiles visible, e.g. χ^2 has a maximum of 5.9, whereas 2d worst goes up to 17.0.

We compare the robustness of estimation accuracy with competing estimators, the IPF and 1d estimators, in Figure 8.9(b). For better readability we plot only the



(a) graph construction strategies (b) varying estimators

Figure 8.9: Accuracy of density estimation

DensEst χ^2 method in these figures. As we can see, the most costly estimator, the IPF strategy shows the lowest deviation and also the smallest variance. The much more efficient χ^2 strategy, performs only slightly worse, but far better than 1d histograms, that result in a very rough guess with huge variance in the result (a maximal deviation factor of 74). This is clearly not feasible for practical applications.

8.3.3 Density estimation in subspace clustering

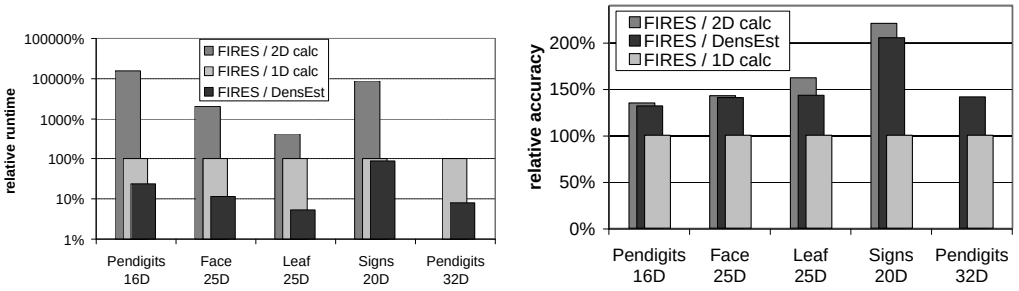
We evaluate the runtime of the subspace clustering algorithm FIRES using three different density estimation approaches: DensEst refers to our new approach with the randomized tree approach that has shown to be both efficient and accurate. It is compared to the $1d\ calc$ that corresponds to the original FIRES approach that is based on 1d histograms. Additionally, we include a $2d\ calc$ version for comparison that is an exact computation of 2d histograms. As we can see in Figure 8.10(a) for several different real world data sets, DensEst outperforms the exact calculations, especially those in two dimensions. Using a logarithmic scale we see that DensEst outperforms the costly exact 2d calculation by orders of magnitude. Note that we have tried to find optimal parameters for each data set as given in Tab. 8.2.

We also study the accuracy of FIRES using these density estimators. One common measure in recent studies of subspace clustering is to measure the accuracy of classifiers (e.g. C4.5 decision tree) built on the detected patterns [BZ07]. A high accuracy indicates that the found subspace clustering is a good generalization of the underlying data distribution.

Figure 8.10(b) shows the resulting accuracy for the same real world data sets as in the previous experiment. As we can see, DensEst greatly improves the accuracy values of the original 1d histograms and is often even close to the full 2d histogram calculation. Thus the estimates based on 2d histograms are very close to the accuracy

	Pen 16D	Face 25D	Leaf 25D	Signs 20D	Pen 32D
ε	0.12	1.5	1.5	3	0.12
$minPoints$	8	10	10	8	8
k	15	25	6	20	25
μ	14	24	5	14	24
$minClu$	1	1	1	1	1

Table 8.2: Parameter setting for FIRES



(a) runtime (b) accuracy

Figure 8.10: Varying estimators in the subspace clustering approach FIRES

of the full calculation with far higher runtimes as seen in the previous experiment.

The analysis using the F1 measure validates the findings of the previous experiments. As depicted in Figure 8.11, DensEst is an estimator of very high quality, especially with respect to its low runtime.

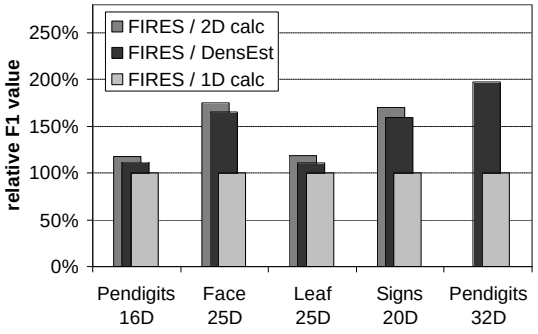


Figure 8.11: F1 values of estimators in FIRES

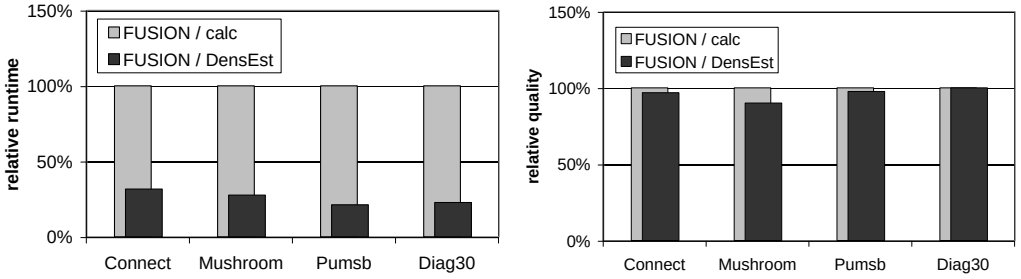
8.3.4 Density estimation in pattern fusion

We also integrated our DensEst technique into the Pattern Fusion algorithm for frequent itemset mining and evaluated it on several real world data sets. Figure 8.12(a) shows that the runtime of our density estimator is only a small fraction of the runtime

required for exact computation in the original Pattern Fusion algorithm (denoted as calc in the figure), where the set of similar low dimensional patterns is required for exact calculation of high dimensional candidates. Note that, we tried to find optimal parameters for each dataset as given in Tab. 8.3.

	Connect	Mushro.	Pumsb	Diag30
min. support	30%	5%	75%	50%
initial pool	1000	20000	1000	1000
initial length		4		
τ		0.5		
K		50		

Table 8.3: Parameter setting for FUSION



(a) runtime (b) quality

Figure 8.12: Varying estimators in Pattern Fusion

The quality measure for comparing the Pattern Fusion estimators follows the approach used by its authors [ZYH+07]. As we can see in Figure 8.12(b), also in frequent itemset mining our DensEst approach performs extremely well on all real world data sets. It is a very tight approximation of the full calculation in the original Pattern Fusion technique, even though it is much faster to compute.

8.4 Enhancements by density estimation of subspace regions

Scaling up data mining algorithms via direct "jumps" to high dimensional candidate patterns requires fast and sound estimates of the potential of each candidate. Our novel DensEst technique provides such estimates very accurately based on 2d histograms and their correlations. These histograms and the resulting estimate can be efficiently computed, even for single selected candidates. Our experiments demonstrate the quality and efficiency of DensEst for both subspace clustering and frequent itemset mining on real world data.

By our DensEst approach, we enable the general processing schema of jumping to high dimensional subspace clusters. This processing schema is the key for efficient subspace clustering. In the following chapter we base on the jump idea and develop a technique which steers the subspace cluster search directly to the most promising subspace regions. In addition to the efficient estimation, our novel steering ensures high quality clusters to be detected first. Overall, this yields a significant efficiency improvement for our high quality subspace clustering models.

Chapter 9

Efficient subspace clustering of relevant subspace candidates only

In this chapter we propose steering of subspace cluster search for efficient density-based subspace clustering. In contrast to our approaches in the previous two chapters, we propose an efficient approximative algorithm for our optimization model described in Chapter 3. Thus, we base on a model defining high quality subspace clusters while its computation is proven to be NP-hard. Our novel steering overcomes efficiency problems caused by database access for density-based clustering, the exhaustive search space of arbitrary subspaces and the optimization of the final result set. We systematically choose the most promising dense subspace regions to be clustered first without having to process large parts of the remaining search space. The selection process for these dense regions is designed such that density information and feedback from progressively detected clusters, steer our subspace clustering approach to only few but high quality subspace clusters.

In addition to our density estimation technique (cf. Chapter 8), this steering can be seen as a general processing schema for efficient subspace clustering. It achieves high quality results by computing an approximation of our relevant subspace clustering as described in Definition 13, but also ensures efficient computation by our steered processing. It overcomes major drawbacks of traditional processing methods. The general idea of jumping directly to the most promising subspace regions is the key for scalability of subspace clustering algorithms. However, the overall quality is highly depending on the selection of these jumps. Thus, in this chapter we propose a novel steering based on an informed selection of jumps. In addition to the previous efficiency improvements, our steering is based on our relevance model for selecting most relevant subspace clusters first. In thorough experiments on real and synthetic databases we show that our steering yields substantial efficiency improvements over existing subspace clustering approaches, while maintaining high quality results.

9.1 Motivation and comparison with related work

A major challenge for subspace clustering approaches is efficiency as they have to cope with a search space of subspace projections that increases exponentially with the

number of dimensions. As summarized in two surveys [PHL04, KKZ09] and compared in our evaluation study in Chapter 10, different subspace clustering approaches have been proposed: On the one hand, efficient algorithms showing comparatively poor quality results, and on the other hand complex models showing good clustering quality but poor runtime behavior.

First, bottom-up approaches such as CLIQUE search almost all subspaces [AGGR98]. Both, traditional breadth-first and our depth-first processing require exhaustive processing of subspace dimensions. Thus, none of the proposed techniques scales w.r.t. the dimensionality. Furthermore, the result sets become overwhelmingly large, resulting in low quality clusterings even for our local pairwise redundancy elimination model proposed in Chapter 2.

Second, projected clustering like PROCLUS partition the database into clusters in subspace projections [AWY+99, PJAM02, YM03]. In enhanced approaches [PJAM02, YM03], randomization is used to speed up the search for projected clusters for an overall efficient processing. However, as major drawback of the underlying models, all of these efficient partitioning approaches cannot detect subspace clusters that share objects.

Third, subspace selection approaches have been proposed. Heuristics for selecting subspaces prior to clustering have been proposed [CFZ99, KKKW03]. However, clusters usually only exist in some parts of a subspace with noisy data in the residual subspace. Such clusters cannot be detected by these methods. Other approaches use density estimation of regions in subspaces as described in Chapter 8. While this is a promising first step in terms of efficiency, it provides only low quality due to the limited information on just one or two dimensions. Furthermore, the overall quality of the computed subspace clusters is relatively poor due to the underlying subspace clustering model [KKRW05].

In contrast to these related approaches, we propose an efficient algorithm with high quality results based on our optimization model (cf. Chapter 3). We follow the density-based clustering paradigm which shows high quality results, especially due to our global optimization of most relevant subspace clusters. However, existing processing paradigms such as the widely used bottom-up processing do not scale to subspace clustering in high dimensional databases. We illustrate the general drawback of bottom-up subspace clustering approaches in Figure 9.1. Searching in a bottom-up processing, they perform an exhaustive search of almost all subspace projections by progressing from k-dimensional to $k+1$-dimensional subspaces. In each of these subspaces they have to search for dense regions, which by itself is a cost intensive task. Since indexing objects w.r.t. exponentially many subspace representations is generally not beneficial, costly database scans are required for the actual density computation. Moreover, for our complex optimization models we have also to cope with a huge set of possible subspace clusters. Choosing an optimal result set out of these subspace clusters is proven to be an NP-hard problem.

In this work, we propose a novel processing scheme for efficient density-based subspace clustering. In contrast to the bottom-up scheme, we propose to steer subspace clustering directly to the most promising high dimensional subspace regions.

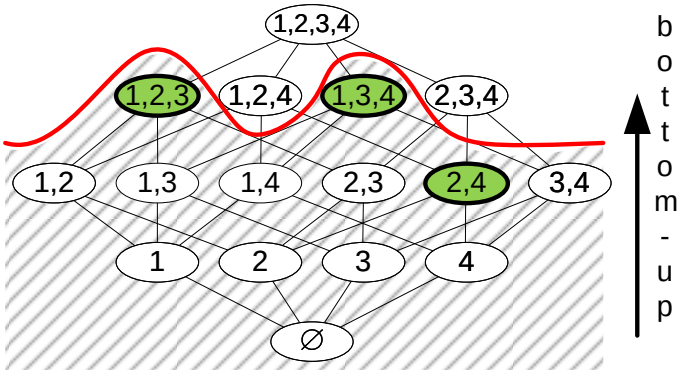


Figure 9.1: Bottom-up processing of search space

Redundant low dimensional projections thus do not have to be generated. Our steering scheme tackles the major efficiency challenges for an efficient processing of the search space:

1. We exclude large low dimensional parts of the search space as we jump directly to high dimensional subspaces.
2. We use efficient density estimates for choosing the most promising regions to jump to.
3. As we progressively detect clusters, we steer to other regions that are not yet represented in the result.

Our steering scheme can be likened to "informed" search, where we actively gather information and account for progressively detected results. Obviously, informed search is far more efficient than "blind" ("uninformed") breadth-first or depth-first search [RN03].

9.2 Formalization of efficiency challenges

In this section, we present our novel steering schema for efficient detection of density-based subspace clusters. Our approach consists of a best-first search which gathers information in lower dimensional spaces and from progressively detected subspace clusters. We start by formalizing the efficiency challenges derived out of density-based subspace clustering.

Given a database DB subspace clustering searches for clusters in subspace projections of the dimension set DIM. Based on our density-based cluster definition given in Chapter 2 and our optimization model proposed in Chapter 3 we search for subspace clusters (O, S) as a set of objects $O \subseteq DB$ which forms a dense region in a subspace $S \subseteq DIM$. Furthermore, according to our relevance model, we search for a non-redundant subspace clustering which is a representative subset $R \subseteq ALL$ of all possible subspace clusters. Overall, density-based subspace clustering is inherently complex due to three main issues related to these subset properties $S \subseteq DIM, O \subseteq DB, R \subseteq ALL$. We formalize each of these challenges in the following.

Search Space of Subspace Clustering First, subspace clustering aims to identify clusters in any possible subspace $S \subseteq DIM$. Thus, it is exponential in the number of dimensions. This leads to high runtimes since subspace clustering targets high dimensional spaces by definition. For many applications, these runtimes even mean that the computation is just not possible in a reasonable time span.

Efficiency Challenge 1. Exponential Search Space
The number of subspaces (i.e., the search space) is exponential in the number of dimensions:

$$|\{S \mid S \subseteq DIM\}| = 2^{|DIM|}$$

State-of-the-art methods use "blind" non-informed search to tackle this exponential complexity. Bottom-up processing in e.g. breadth-first order uses monotonicity to prune the search space [AGGR98, KKK04]. Monotonicity means that a subspace cluster induces clusters in all of its lower dimensional projections:

$$\begin{aligned} &(O, S) \text{ is a subspace cluster} \\ \Rightarrow &\forall S' \subseteq S : (O, S') \text{ is a subspace cluster.} \end{aligned}$$

The negation is used for pruning in bottom-up processing. If a set of objects O is not clustered in subspace S', then prune all its higher dimensional projections S. However, this requires generating the vast majority of low dimensional projections. For a d-dimensional database, the runtime complexity can only be reduced from 2^d to 2^k where k is the maximal dimensionality of any hidden cluster. In many real world applications, 2^k results in unacceptable runtimes. For example, runtimes of 7 days have been reported for 50-dimensional databases, and exponential increase w.r.t. maximal cluster dimensionality [KKK04]. Poor runtime behavior is also confirmed by our evaluation study in the following chapter.

Density-Based Clustering Second, density-based clustering requires computing the density of all objects, $density(o) = |\{p \in DB \mid dist_S(o, p) \leq \varepsilon\}|$, i.e. the objects within a certain distance in subspace S contribute to the density of this object. Consequently, for each object, the set of its neighboring objects needs to be determined and evaluated. This results in repeated database scans for density computation, and in a computational complexity that is quadratic in the number of objects in the database.

Efficiency Challenge 2. Complexity of Density Computation
For each subspace S, computing the density is quadratic in the database size

$$\textit{Density-based clustering one subspace} \in O(|DB|^2)$$

To efficiently determine ε-neighborhoods for each object in a single (full) space, index structures may be used as in [EKSX96] for efficiency improvement. However, in subspace clustering, one would need index support for any subspace projection. To the best of our knowledge there exists no such index structure, while building an

exponential number of traditional indexes is not efficient. Thus, only one-dimensional index structures with intersection operations (inverted lists) are applicable [KKK04]. If an index for multiple projections becomes available in future research, our approach could benefit from it as well.

Choice of Final Result Set And lastly, computation of the final result $R \subseteq ALL$ is a challenging task. Given the large number of subspace clusters ALL, selecting a subset of few, but representative subspace clusters amounts to exponentially many options. Out of these options a high quality result has to ensure that each detected cluster contributes novel knowledge about the hidden structure of the data. However, this choice poses a major challenge, especially as the size of ALL increases with the number of subspaces, i.e. exponentially with the number of dimensions (cf. Efficiency Challenge 1).

Efficiency Challenge 3. Exponentially many possible result sets
There are exponentially many subsets R in the set of all subspace clusters ALL:

$$|\{R \mid R \subseteq ALL\}| = 2^{|ALL|}$$

In addition, for the choice of R two clustering properties are of importance. First, in clustering one aims at covering almost all objects by at least one detected cluster, excluding only noisy objects which remain uncovered. And second, one has to ensure that only few but interesting high dimensional subspace clusters are included. High dimensional clusters are preferred as they represent many low dimensional subspace projections and provide the most information about the hidden structures in the data. Thus, a subspace clustering $R = \{C_1, \ldots, C_n\}$, has to maximize the coverage $\max_R Cov(R) = \max_R |\bigcup O_i|$ using few high dimensional subspace clusters. In general, we have shown in Section 3.2.4 that such an optimization of relevant subspace clusters is an NP-hard problem.

Summing up the challenges, for density-based subspace clustering, efficient algorithms are of crucial importance given the size and dimensionality of most practical databases.

9.3 Efficient subspace processing

In this section, we present our basic idea for efficient detection of high dimensional subspace clusters.

9.3.1 Overview of our solution

We propose a steering of the subspace cluster search through the subspace lattice, with intelligent information gathering and learning strategies as we continuously tune in on the best subspace cluster candidates.

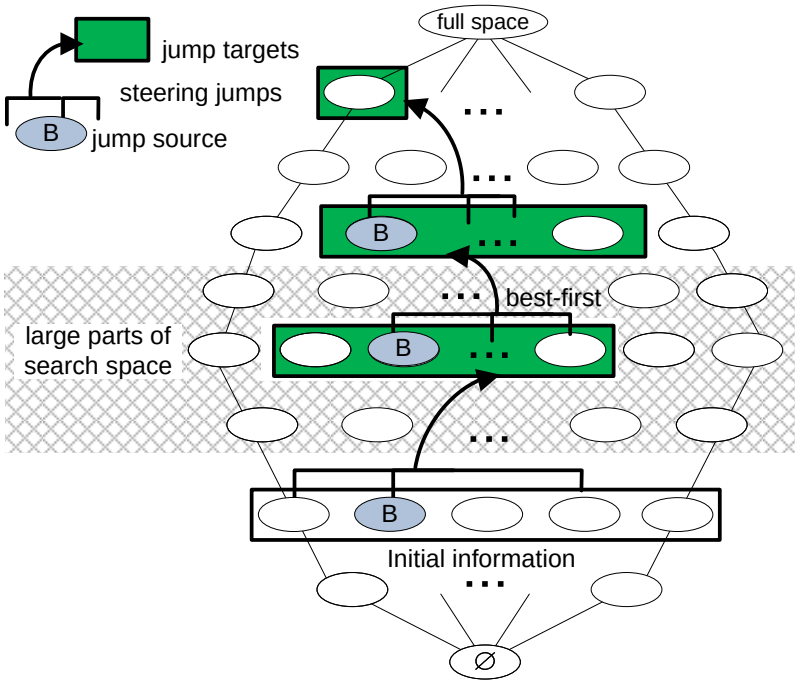


Figure 9.2: Multi-jumps in only few subspaces

Steering the Search The core idea is that we do not need to slowly search low dimensional subspaces in order to work our way up to the high dimensional subspaces. Instead, we observe that any high dimensional subspace cluster is very likely reflected in many low dimensional projections. While this is the drawback for bottom-up approaches, we use this effect to combine low dimensional information for direct *multi-jumps*. Multiple jumps can be seen as a generalization of our single jump using density estimation proposed in the previous chapter.

As depicted in Figure 9.2, we only use the gathered information about some low dimensional subspace regions (*jump sources*) to choose the next candidates (*jump targets*). Thus we jump through the search space without clustering subspaces between jump source and jump targets. Clearly, this is far more efficient. Using the information obtained in higher dimensions to jump even further (multi-jumps), our approach can be likened to a best-first search, where progressively obtained search results steer the algorithm to future jump targets.

Thus, we tackle the exponential search space problem (cf. Efficiency Challenge 1). The main issue for efficient and high quality jumps is choosing the jump targets. We discuss our systematic choice in Section 9.3.3.

Information Gathering and Maintenance To start and run the multi-jump process, we need to gather and maintain information on promising subspace regions. While jumping through subspaces, we gather information about density of subspace regions on multiple density granularity levels. Figure 9.3 illustrates this idea as a pyramid: Level 0 at the bottom of the pyramid consists of rough density estimates of subspaces regions, Level 1 approximates subspace clusters as hyperrectangles in

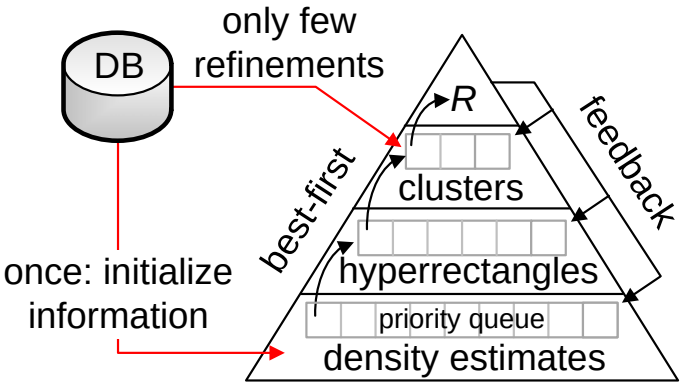


Figure 9.3: Information gathering pyramid

subspaces, and Level 2 at the top of the pyramid maintains subspace clusters for selection. On each level we maintain one priority queue storing density information which represent promising clusters in the already processed subspaces. We sort these queues based on feedback from the result set R and thus ensure to process most promising candidates in best-first order. Furthermore, the two lower levels provide efficient filtering mechanisms which ensure that only those regions will be refined that are expected to lead to promising subspace clusters (cf. Efficiency Challenge 2). Hence, our technique requires database access only once for initialization of density estimates on Level 0 and for refinement of the few candidates on Level 2. The efficient filtering techniques are studied in Section 9.3.2.

Cluster Selection for the Result Set Finally, we have to efficiently compute the final clustering result as a set of density-based subspace clusters. As computation of all subspace clusters is clearly not efficient, we build the result set through best-first selection of subspace clusters. For best-first selection we steer our approach based on the cluster's dimensionality and the set of detected clusters R. In an iterative process, we include those subspace clusters that are rather high dimensional, and which contain new objects not yet covered by R. Hence, we achieve a high quality result set as each cluster in R has a contribution to the extracted knowledge. For efficient processing, thus addressing Efficiency Challenge 3, our feedback mechanism sorts the queues in our pyramid and steers our jumps (cf. Section 9.3.4), which ensures to process and output the most promising clusters first.

Overall algorithm Our algorithm uses the pyramid architecture with information on density estimates, approximate hyperrectangle representations, and progressively computed subspace clusters, to steer the algorithm in a best-first manner through the search space. We progressively select those subspace clusters which contribute new (uncovered) information to the final subspace clustering.

An overview is given in Algorithm 7. First, we initialize our information pyramid by estimating the data distribution from low dimensional (initial) subspaces. This gives us the basis for selecting the best candidate from the pyramid. Depending on

the density granularity level of this candidate, we either refine this candidate (Level 0 and Level 1) and insert the result into the next higher level priority queue, or if it is a subspace cluster (Level 2), we insert it into the result set. On the middle level, we perform multi-jumps to higher dimensional subspaces. Note that since each level maintains its information in a priority queue, we select the most promising candidates for the next step in the search process until the database is represented (covered) entirely, except for possible noise objects.

Algorithm 7 Steering Subspace Cluster Search

```
1  R = ∅ ;                                                    /* initialization of result set */
2  pyramid = initialize information gathering ;               /* create initial information on Level 0 */
3  while ∃ candidate (O, S) with new (uncovered) objects do   /* detect next subspace cluster */
4      bestCandidate = pyramid.getBestCandidate() ;           /* most promising cluster candidate */
5      if bestCandidate on Level 0 then                       /* density estimate information */
6          pyramid.add(bestCandidate.refine(), Level 1) ;     /* refine and add on next level */
7      else if bestCandidate on Level 1 then                  /* hyperrectangle information */
8          targets = steeredJumps(pyramid);                   /* steering subspace cluster search */
9          pyramid.addAll(targets, Level 0);                  /* include novel knowledge on Level 0 */
10         pyramid.add(bestCandidate.refine(), Level 2) ;     /* refine and add on next level */
11     else if bestCandidate on Level 2 then                  /* density-based subspace cluster */
12         R = R ∪ {(O, S)} ;                                 /* add most promising subspace cluster */
13         pyramid.update(R);                                 /* steer by sorting queues based on new coverage of R */
```

9.3.2 Information gathering

The accuracy of our approach clearly depends on the quality of the jumps, i.e. on gathering sufficient information for steering the algorithm to the most promising subspace clusters. We start by describing the rough approximation at the basis of the pyramid. On the lowest level in the pyramid architecture, the goal is to efficiently determine promising regions in different subspace projections. For density-based subspace clustering, we therefore need to quickly assess the density of regions. Obviously, a region that is not dense cannot contain a density-based subspace cluster and can be excluded from further processing. Thus, our first filter step uses the data distribution represented by histograms for density estimations of subspace regions. Histograms can simply be seen as a discretization of data. While more enhanced discretization techniques might be used here, we focus more on the general processing using this discretized information. Thus, we define regions as hypercubes according to a simple regular grid structure, formed by dividing up each dimension d_i into intervals $j_{d_i} \in Int(d_i)$:

Definition 29. Subspace histogram
A histogram h_S in a subspace $S = \{d_1, \ldots, d_s\}$ is defined as

$$h_S : Int(d_1) \times \ldots \times Int(d_s) \to \mathbb{R}$$

$$(j_{d_1}, \ldots, j_{d_s}) \mapsto |Obj(H)|, \;\; H = [(d_1, j_{d_1}), \ldots, (d_s, j_{d_s})]$$

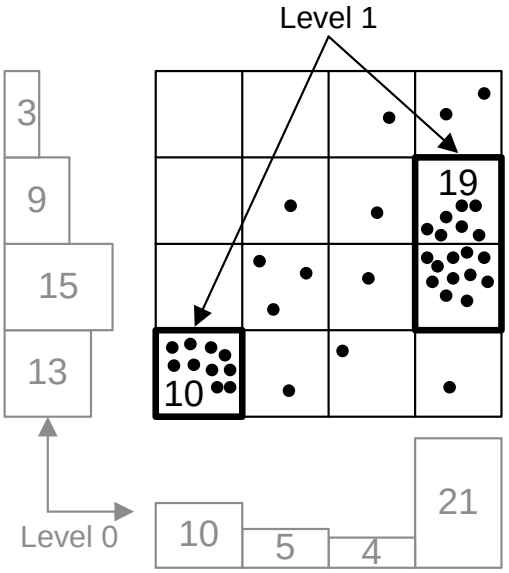


Figure 9.4: Gathering filter information

A region H is uniquely identified via its interval identifiers $j_{d_i} \in Int(d_i)$ per dimension $d_i \in S$. The histogram is simply the number of objects in this region $|Obj(H)|$ as illustrated in Figure 9.4.

A histogram is a simple technique for estimating densities on the lowest level of our pyramid (Level 0 in Fig. 9.3). For example, in Figure 9.4, the region with a value of 21 in the histogram below, and a value of 15 at the left is more likely to contain a subspace cluster than the region with 4 below and 3 to the left. For efficient, but accurate density estimation, we use density estimation based on 2-dimensional histograms as proposed in Chapter 8. Please note that it is possible to estimate the density for individual regions, without having to estimate the density for the entire subspace.

While the density estimates are very efficiently computable, they give only a rough approximation of potential subspace cluster regions. In order to steer our algorithm effectively through the search space, we use a much tighter approximation of subspace clusters through hyperrectangles on the next level of the pyramid (Level 1). Each such hyperrectangle is a region in a subspace that contains many objects and therefore might represent a density-based subspace cluster. It is the core information for steering the algorithm through multi-jumps, as will be discussed in Section 9.3.3.

Hyperrectangles are obtained through discretization of the search space. Discretization is a well-known method for speeding up subspace clustering within a given subspace [AGGR98, SZ04, NGC01] as also used in our multi-step approach in Chapter 5. Figure 9.4 illustrates the discretization process in a 2-dimensional space with two clusters. Instead of processing objects from the database, discretization simply processes discretized cell counts. For each cell, the object count is maintained. Density-based subspace clusters are represented in the discretized space by adjacent cells showing a high number of contained objects. In the figure, hyperrectangles (cells surrounded by a thick line) represent the underlying density-based subspace clusters.

In our approach, these hyperrectangles in a discretized subspace represent subspace cluster candidates:

Definition 30. Hyperrectangle
A hyperrectangle H in subspace $S = \{d_1, \ldots, d_s\}$ is specified by a tuple $[(d_1, low_1 - up_1), \ldots, (d_s, low_s - up_s)]$ of lower low_i and upper up_i bound interval identifiers in dimension d_i.

We write $H = [(1, 4), (2, 3-5), (3, 6)]$ if upper and lower bounds in a dimension are identical, i.e. short for $H = [(1, 4-4), (2, 3-5), (3, 6-6)]$. We denote the dimensions that are contained in H as $Dim(H)$, and the corresponding intervals as $Int(H, d_i)$.

If a region represented as a hyperrectangle is processed further, the next level uses density-based subspace clustering (as proposed in Chapter 5) to refine it. In the pyramid illustration in Figure 9.3, this is the top level information (Level 2) before a cluster is output as a result.

9.3.3 Steering through the search space

In this section, we describe how, based on the information we have gathered in the priority queues, we steer the algorithm to the most promising search space regions in a best-first manner. The general idea is derived from the property that high dimensional subspace clusters show up in their lower dimensional projections. Thus, low dimensional information (jump sources) is sufficient for finding dense regions (jump targets) in higher dimensional subspaces.

Initialization phase To start the algorithm, we need to provide initial information as the first jump sources. Please note that in contrast to traditional bottom-up algorithms, our approach only requires a few initial cluster candidates for starting the multi-jump process. This information indicates the first jump targets and with multi-jumps will lead us to high dimensional subspace clusters. Hence, a sample of the subspaces in the initial 2-dimensional spaces is sufficient to generate initial jump information. The parameter *sampling* sets the number of initially processed subspaces. As an extreme, *sampling* = 100% would correspond to searching all subspaces of the initial dimensionality. We evaluate the robustness of the clustering result w.r.t. this parameter in our experiments.

Multi-jump phase The goal of the jump phase is to generate new higher dimensional hyperrectangles based on already generated lower dimensional hyperrectangles. To reach high dimensional subspaces via jumps information from the very low dimensional spaces might not be sufficient, e.g. jumping from 2-dimensional regions directly to 20-dimensional regions is almost random. Thus, we introduce multiple jumps with successive information gathering (cf. Figure 9.2). Using this intermediate information leads to better steering of our approach, enhanced jumps, and thus, to better clustering quality.

Technically, we gather information as subspace cluster candidates in arbitrary subspace projections. We combine this information to higher dimensional candidates based on the observation that dimensions occurring frequently in our cluster candidates are indicators of dense high dimensional subspaces. Steering the jumps to most promissing subspaces is done by determining for a *jump source*, i.e. a promising hyperrectangle at the top of the priority queue, based on the available *jump information*, i.e. other gathered hyperrectangles, a ranking of potential *jump targets*, i.e. hyperrectangles which should be combined with the jump source in order to generate new hyperrectangles in higher dimensionalities.

We note the following jump terminology:

- The highest ranked hyperrectangle B (on which the jump is based) is called the jump source.
- The set $\mathcal{I}$ of remaining hyperrectangles (in the priority queue) is denoted as jump information.
- A jump target T is a hyperrectangle selected for a jump in combination with a jump source B. T does not contain any dimensions which are already part of the jump source: $Dim(T) \cap Dim(B) = \emptyset$.

In a running example for this section we assume, a given jump source $[(1,2), (2,2), (4,2)]$ as the most promising hyperrectangle B. Derived out of the following hyperrectangles in the priority queue, we obtain jump information that steers us to $[(5,2), (7,2), (6,2), (11,2)]$ as jump target T. In the following, we first describe the jump procedure combining these two hyperrectangles to form a new candidate in a higher dimensional subspace, and second, we discuss details about the actual selection of jump target out of the given priority queue.

In general, selected jump targets are combined with the jump source to form new candidates based on the following definition:

Definition 31. Combination of hyperrectangles
Given two hyperrectangles H_1 and H_2 which do not share any dimensions, i.e. $Dim(H_1) \cap Dim(H_2) = \emptyset$. Their combination is a higher dimensional hyperrectangle which contains the dimensions and intervals of the two originating hyperrectangles:

- $Dim(H) = Dim(H_1) \cup Dim(H_2)$
- $\forall d \in Dim(H_1) : Int(H_1, d) = Int(H, d)$
- $\forall d \in Dim(H_2) : Int(H_2, d) = Int(H, d)$

If the first hyperrectangle from the Level 1 priority queue has been chosen as jump source, then it is combined with jump targets. It might be refined to yield a new subspace cluster. This leads to the generation of new and higher dimensional subspace clusters. Including these candidates in our gathered information, they either

can be chosen as the next jump source or used as jump information for other jump sources. Overall, inclusion of new candidates increases the available information and leads to higher quality of our multi-jumps in even higher dimensional subspaces.

Determining jump targets Once we have selected the first hyperrectangle from the priority queue in our pyramid as the jump source, we need to find jump targets. This requires identifying dimensions and intervals from the currently available jump information in the rest of the queue for possible combination. Obviously, dimensions already included in the jump source do not lead to new jump targets, and neither do combinations of different intervals in the same subspace. We refer to valid combinations of hyperrectangles in the jump information as *jump indicators*. As an indicator we describe a subset of dimensions and intervals that show up as frequent combination in the available jump information. The higher the frequency of an indicator, the more probable it belongs to a high dimensional subspace cluster. These indicators are the most promising hyperrectangles to be added to the selected jump source, and thus, are used for determining the jump targets.

Formally, jump indicators are described as:

Definition 32. Jump indicator
A jump indicator P of length k is a hyperrectangle, which does not share any common dimensions with the jump source B ($Dim(P) \cap Dim(B) = \emptyset$) and $Dim(P) = k$.

In our example, the jump target is derived out of a ranking of indicators $(5, 2)$, $(7, 2), (6, 2), (11, 2), \ldots$ sorted by their frequency in the jump information. Out of this ranking we combine the best indicators for our jump targets. In general, we rank jump indicators by their potential impact, i.e. how "helpful" they are in leading to subspace clusters for our result set. Deriving the impact is discussed in the following.

Combining jump indicators In the final step, the jump targets have to be determined from the different jump indicators. We propose two efficient methods. Their basis is a ranking of the k-length jump indicators. In the following, we illustrate this process assuming $k = 1$ for simplicity of presentation. The extension to higher values of k is straightforward.

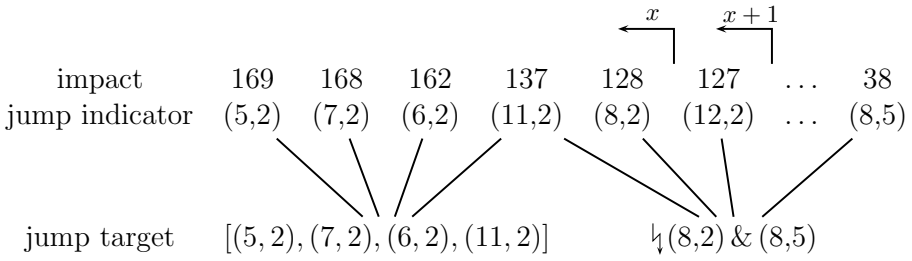


Figure 9.5: Generating targets

Figure 9.5 gives an example of some one-dimensional jump indicators (tuples of dimensions and interval identifiers). As shown in the figure, the combination of four

one-dimensional jump indicators yields a jump target of dimensionality four. Please note, that some combinations are not valid, since they would result in disjoint intervals in the same dimension (lower right in the figure). Thus, for jumps of a certain size, we construct jump targets by combining smaller k-dimensional indicators. These potential jump targets should be ranked, such that we always process the most interesting combinations first.

In the *deterministic combination* method, our idea is to include jump indicators up to a fixed position x in the ranking. In Figure 9.5, only jump indicators up to the first arrow labeled "x" are taken into consideration. Jump targets are constructed only from these jump indicators. Once those jump targets have been generated, the position of x is incremented by one, and new jump targets are generated. In the figure, this is illustrated by the arrow one position to the right ($x + 1$).

In the previous method, the jump indicators are included in the combination process up to a fixed point, regardless of their differences in terms of interestingness. We propose a second method for selecting jump indicators for combination which relies explicitly on this interestingness measure. The *randomized combination* method draws weighted samples on the set of all jump indicators, where the probability of selecting any jump indicator corresponds to its interestingness value. Thus, two jump indicators with the same interestingness would be chosen with identical probability, whereas a jump indicator with double the interestingness would also be drawn with a probability that is twice as high. All randomly drawn jump indicators are combined to jump targets as before, taking only valid combinations.

As main parameter for generation of jump targets we use $generatedClusters$. This parameter sets the number of subspace clusters that are generated from a given jump source. These subspace clusters are added as candidates to the priority queue on Level 0 of our pyramid. Since the approximations do not all necessarily result in a subspace cluster, the parameter only counts "successful" jumps. To avoid a worst-case, i.e. a long series of non-successful jumps that do not generate any subspace clusters, we cut off the generation for this hyperrectangle eventually. We evaluate the two combination methods experimentally in Section 9.4.

Impact of jump indicators Obviously, the crucial part in determining jump targets lies in the ranking of the jump indicators based on their impact. As mentioned above, we know that any high dimensional subspace cluster is likely to be reflected in many different low dimensional projections. We use this observation to calculate the impact. If a jump indicator is part of several currently known subspace regions, this suggests that it is a projection of a higher dimensional cluster. In order to extend the jump source, we determine the similarity to the jump source for any hyperrectangle in the jump information set:

Definition 33. Impact of jump indicators
Let P be a jump indicator, B the jump source and $\mathcal{I}$ the jump information. The impact of P is the sum of similarities of all hyperrectangles $H_i \in \mathcal{I}$ in which P is

contained to the jump source B

$$\sum_{H_i \in \mathcal{I},\ P_{in}\ H_i} cs(B, H_i)$$

An obvious question to ask is how to obtain the jump indicators and their impact. We propose an efficient way of generating all possible jump indicators along with their impact. We simply transform the problem of combining hyperrectangles (given by their intervals in the respective dimensions) into a problem of (weighted) itemset mining [HK01, HPY00]. A k-dimensional jump indicator can be mapped to a k-itemset, i.e. a set of k items that occur in a transaction. The (weighted) support, i.e. the number of occurrences of this itemset, corresponds to the impact. In itemset mining, efficient algorithms exist [HPY00], which we use to compute the interestingness, and the top-m-ranked results.

An example is given in Figure 9.1: at the top, the jump source is given as a set of tuples of dimensions and interval identifiers. To determine the desired jump indicators, the source can be combined with the remaining hyperrectangles in the first column. The second column notes dimensions and intervals which could be used to extend the source (mapped to transactions). To compute the impact with respect to a jump indicator, e.g. [(5, 2)], we sum up the similarities to the source as the (weighted) support of all transactions that contain the jump indicator. So, the first transaction contributes to the support, but not the second, etc.

jump source: $[(1,2),(2,2),(4,2)]$, jump indicator $[(5,2)]$

jump information (hyperrectangle I_i)	transactions (potential extension)	similarities cs of I_i	support for $[(5,2)]$
$[(1,2),(2,2),(5,2)]$	$\{(5,2)\}$	**2**	+2
$[(2,2),(4,2),(7,2)]$	$\{(7,2)\}$	**2**	
$[(1,2),(5,2),(6,2)]$	$\{(5,2),(6,2)\}$	**1**	+1
$[(1,2),(5,2),(7,2)]$	$\{(5,2),(7,2)\}$	**1**	+1
⋮	⋮	⋮	⋮

Table 9.1: Mapping to itemset mining

Please note that even though the above discussion assumes that each hyperrectangle is given by one interval per dimension, this can easily be extended to ranges of intervals. For example, an entry $(1, 3-5)$ would be mapped to items $(1,3),(1,4),(1,5)$.

Target selection based on cluster similarity Calculation of similarities between subspace clusters is an important aspect for the impact of jump indicators. The higher the similarity $cs(B, H_i)$ of a hyperrectangle H_i, the more interesting this hyperrectangle is with respect to the jump source B.

The function cs (cluster similarity) assigns a higher weight to those hyperrectangles which are more similar to one another. A high similarity indicates that both

hyperrectangles might be a projection of the same high dimensional subspace cluster. Thus, this jump indicator should be prioritized. The goal in determining the cluster similarity *cs* is that the more dimensions are shared by the jump source and another hyperrectangle, the more similar they should be. Also, we need to take into account in which intervals they overlap. Clearly, if the intervals are disjoint, the two hyperrectangles do not represent the same higher dimensional subspace cluster.

In Figure 9.6, two hyperrectangles are depicted in their two-dimensional projections. Both of them contain dimensions 1 and 2, but since they do not share intervals in dimension 2, they do not cover the same area in the search space. Thus, they do not represent the same cluster, and should have a similarity value of 0.

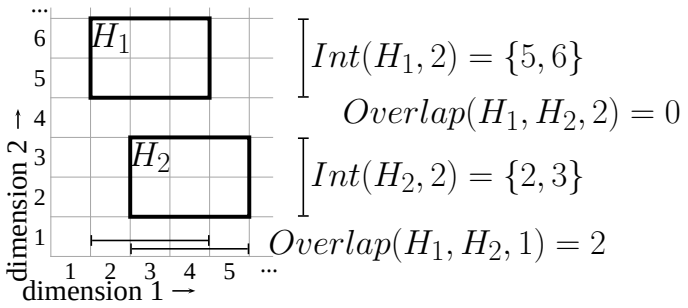


Figure 9.6: Hyperrectangles in the same subspace with no overlap (disjoint intervals in dimension 2)

Thus, only overlapping hyperrectangles have positive similarity values. We define the number of overlapping intervals of two hyperrectangles H_1 and H_2 in dimension d as

$$Overlap(H_1, H_2, d) = \begin{cases} |Int(H_1, d) \cap Int(H_2, d)| & \text{, if } d \in Dim(H_1) \cap Dim(H_2) \\ 0 & \text{, else} \end{cases}$$

Using just the cardinality of the intersection of the dimensions, e.g. $|Dim(B) \cap Dim(H_i)|$ as the similarity measure, takes the number of dimensions into account in which there is an overlap, but ignores the extent to which they overlap. Hyperrectangles which overlap more (e.g. Fig. 9.7 (a) more than (b)), are much more likely

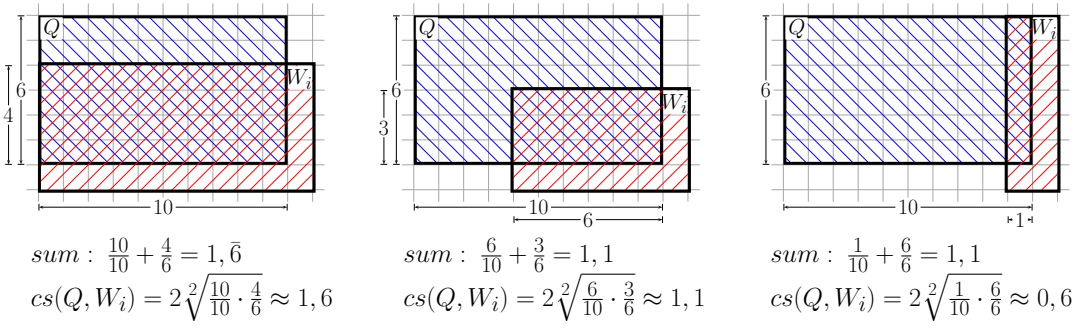


Figure 9.7: Similarity computation

to be reflections of the same higher dimensional subspace cluster, so they should be considered more similar with respect to the function cs.

Considering overlap with respect to the intervals in the jump source by using $Overlap(B, H_i, d)/|Int(B, d)|$, much better reflects the degree to which the two hyperrectangles are thought to reflect the same higher dimensional subspace cluster. The sum over these overlaps $\sum_{d \in Dim(B) \cap Dim(H_i)} (Overlap(B, H_i, d)/|Int(B, d)|)$ conceals small overlaps in one dimension if there is a larger overlap in another. We therefore propose using a product of the overlaps. Consider the example in Figure 9.7: taking the sum of overlaps yields $6/10 + 3/6 = 1.1$ in (b) and also $1/10 + 6/6 = 1.1$ in (c), even though the overlap is considerably larger in (b). Taking the product of the overlapping intervals is e.g. in (c) $1/10 \cdot 6/6 = 1/10$, and in (b) a higher value of $6/10 \cdot 3/6 = 3/10$.

The number of dimensions, however, is underrated: e.g. an overlap of 8/10 per dimension yields $(8/10)^2$ for two dimensions, yet $(8/10)^3$ for three dimensions. To make sure that we still favor those situations where more dimensions overlap, we introduce a normalization factor which intuitively gives us the "average" overlap per dimension. Considering the product, e.g. $1/10 \cdot 6/6 = 1/10$, as a volume in the k-dimensional space, we determine the side length of a hypercube of the same volume, i.e. we compute the kth root of the volume. In the example in (c), this hypercube has a side length of $\sqrt[2]{1/10} \approx 0,32$, whereas (b) has a value of $0,55$. Multiplying this value with the number of overlapping dimensions k is our definition of the cluster similarity:

Definition 34. Cluster Similarity
For a given jump source B, and the hyperrectangles H_i available as jump information, let $|Dim(B) \cap Dim(H_i)| = k$. We define the cluster similarity cs as follows:

$$cs(B, H_i) = \begin{cases} k \cdot \sqrt[k]{\prod_{d \in Dim(B) \cap Dim(H_i)} \frac{Overlap(B,H_i,d)}{|Int(B,d)|}} & , \text{if } Overlap(B, H_i, d) \neq 0 \\ 0 & , \text{else} \end{cases}$$

Thus similarity is the average overlap between jump source and jump information and therefore reflects the degree to which promising new hyperrectangles can be generated through their combination.

9.3.4 Best-first cluster selection

Recall that the goal of subspace clustering is determining the final subset of subspace clusters $R \subseteq ALL$ that are output to the user (cf. Efficiency Challenge 3). To meet the purpose of any knowledge discovery process, a manageable number of patterns should be presented to the user. As a consequence, the subspace clustering literature has seen several approaches which aim at result size reduction. As discussed in the previous chapters they range from removal of redundant lower dimensional clusters in pairwise comparison, to more complex redundancy elimination strategies in our optimization models, and to global selection in a statistical or heuristic manner [MS08].

As we propose in Chapter 3, a subspace clustering should ideally represent almost all objects in the database (except for noise), and it should high dimensional subspace clusters which represent many low dimensional projections. In the following, we provide details about how progressively detected clusters steer the search to novel dense regions that are not yet represented. For efficient subspace clustering based on this best-first processing, we propose two kinds of steering. First, we steer the choice of jump targets and second, we steer the queues on each level of our pyramid.

Steering jump target selection We propose a function *cp* (cluster preference) to measure representation of new information. Subspace clusters which contain objects which are already represented (covered) by the result set contribute little or no new knowledge to the result. By prioritizing hyperrectangles that are not yet covered, we favor those jump targets which are likely to lead to the desired subspace clusters. Unlike *cs*, this second measure *cp* is an unary function that is only applied to the hyperrectangle in the jump information, since the jump source is fixed.

To cover many objects with few subspace clusters, preference should be given with respect to the number of objects in a hyperrectangle, i.e. $|Obj(H)|$. With respect to the currently known subspace clusters available, R and its covered objects $Cov(R)$, we denote the objects in H that are already contained in R as $|Obj(H) \cap Cov(R)|$. We define the preference function such that those hyperrectangles are preferred which contain many objects that are not yet included in other subspace clusters:

Definition 35. Cluster preference
Let R be the current set of already detected clusters. Then we define the cluster preference function cp for a hyperrectangle H as the number of objects in H that are not yet covered:

$$cp(H) = |Obj(H)| - |Obj(H) \cap Cov(R)|$$

Overall, we define the impact of a jump indicator as the product of the cluster similarity and the cluster preference values:

Definition 36. Impact of jump indicators
Let P be a jump indicator, B the jump source and $\mathcal{I}$ the jump information. The impact of P is the sum of $cs(B, H_i) \cdot cp(H_i)$ of all hyperrectangles $H_i \in \mathcal{I}$ in which P is contained.

Thus, the impact takes the similarity between the jump source and the hyperrectangles into account to identify promising higher dimensional regions, while giving preference to those that contain not yet covered objects. The jump indicators selected in this manner then represent many low dimensional projections, and at the same time represent many different objects.

Steering candidate selection Within the pyramid, we steer the selection of subspace cluster candidates. The key idea is to maintain subspace cluster candidates in priority queues on different information levels in our pyramid. We always take the most

promising cluster candidate out of these queues. As described in Section 9.3.1, this candidate affects the final result as it is either refined, induces jumps and finally is output to the clustering result. Thus, priority queues are sorted based on the current set of detected clusters R:

$$C_i \prec C_j \Leftrightarrow |O_i \setminus Cov(R)| \cdot |S_i| \geq |O_j \setminus Cov(R)| \cdot |S_j|$$

Thus, a cluster candidate C_i is queued before C_j iff C_i contributes more objects (O_i) to the result set R than C_j (O_j). In addition to sorting based on coverage, we prioritize high dimensional subspace clusters by including the subspace dimensionality $|S_i|$ and $|S_j|$, respectively. With this priority, we focus on a simplified cluster preference motivated out of our local interestingness as proposed in Section 3.2.5. However, other possible priorities could be easily integrated such as our enhanced local interest proposed in Section 4.3.4. These sorting functions enable a flexible steering depending on e.g. application specific priorities.

This steering is used on all levels of our pyramid, such that on the top level we iteratively include the most promising subspace clusters as final cluster selection. Both proposed cluster preference methods use the coverage of R to steer the next cluster detection. For each new cluster candidate (O, S) and current clustering R, we actively steer the search to subspace regions where clusters with not yet covered objects can be detected. Thus, we add the most promising cluster (O, S) with $|O \setminus Cov(R)| > 0$ ensuring that this cluster contributes at least one object to the current result set. We iteratively include more and more clusters into the result set until maximal coverage is achieved $\forall (O, S) \notin R : |O \setminus Cov(R)| = 0$ and obtain this R as the final result set.

Our novel multi-jump technique has a number of advantages. First, existing approaches require computation of *all* x-dimensional subspace clusters prior to computation of the $x + 1$-dimensional subspace clusters. Our approach only requires few database scans to estimate the data distribution from low dimensional subspaces as initialization to the method. Second, existing approaches proceed bottom-up through the subspaces increasing the dimensionality by at most one. Our approach jumps through the search space and thus prunes large parts of it. And finally, by using multi-jumps we continuously increase the available information to steer to promising subspace regions.

As we will show in the following section, our subspace processing scales well with increasing number of dimensions, while detecting high quality clusters.

9.4 Experiments

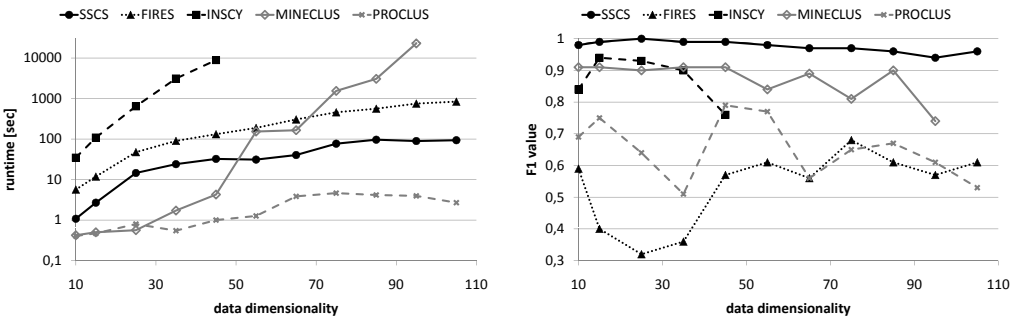
We compare our steering of subspace cluster search (SSCS) approach with recent representatives of different high dimensional subspace or projected clustering paradigms: for complete density-based subspace clustering algorithms, we include our INSCY approach as proposed in Chapter 6; For approximative density-based approaches, state-of-the-art representative is FIRES [KKRW05]; Projected clustering

approaches are represented through the partitioning PROCLUS algorithm [AWY+99] and the recent improvement through MINECLUS [YM03].

Please note that we have optimized parameters for each algorithm on each data set. For a fair comparison of their performance, we report the best quality results for each algorithm and the corresponding runtimes. As measures for clustering quality we use F1 value and accuracy as in the previous chapters. To ensure comparability of runtimes, all implementations are in Java. All experiments were run on the same machine with Intel Quad Core Opteron 2.3 GHz CPUs.

As scalability to large and high dimensional databases is of main importance, we generate large scale synthetic data for scalability experiments. In our synthetic data we ensure that each dimension of a generated high dimensional database is part of at least one hidden subspace cluster. Thus, in contrast to other data generators that simply add noise dimensions to a low dimensional dataset, our high dimensional data is not structurally low dimensional, and no dimension can be excluded as globally irrelevant by the subspace clustering. Following the method in [KKK04] for overlapping density-based subspace clusters, we generate clusters in arbitrary subspaces. We generate databases with different sizes and different dimensionalities, and hide subspace clusters with a dimensionality of 50%, 60% and 80% of the data dimensionality. Furthermore, we scale the dimensionality of the hidden clusters themselves while keeping the data dimensionality fixed.

For evaluation on real world data we use benchmark databases from the UCI archive [AN07], some of them also used in our evaluation study in Chapter 10. Besides the UCI 16-dimensional pendigits data, we use 32 and 48-dimensional variants by interpolation of the available polylines. Furthermore, we use the 50-dimensional Face data set and 50-dimensional Swedish Leafes data set extracted out of sequence data from [KXWR06]. All these data sets are challenging for subspace clustering tasks due to their large number of dimensions. As the true number of clusters is unknown for real data, clustering quality is measured with respect to the class labels.



(a) Runtime (b) Runtime

Figure 9.8: Scalability w.r.t. data dimensionality

9.4.1 Scalability on synthetic data

Our evaluation begins with an analysis of scalability issues on synthetic data. We first scale the number of dimensions, the most important parameter for performance of subspace clustering algorithms. We depict the runtime and quality of the resulting clusters with respect to the dimensionality on a synthetic data set with 10 hidden clusters.

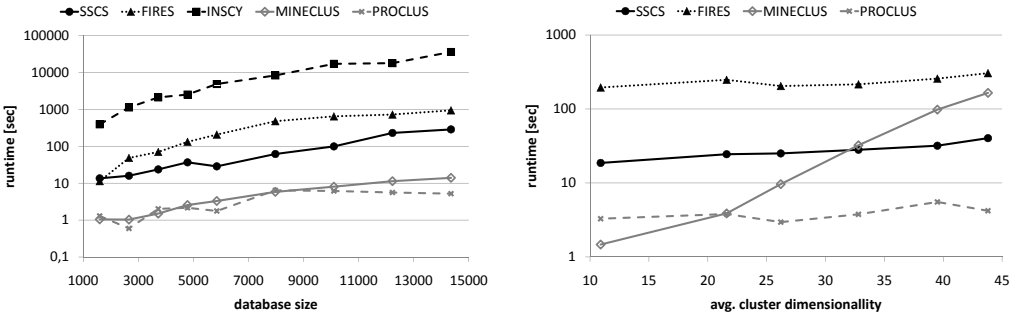
Data Dimensionality First, we confirm the poor performance of bottom-up approaches which process almost all subspaces in a step-by-step fashion. As depicted in Figure 9.8(a), such approaches, here represented by the most recent bottom-up approach for density-based subspace clustering INSCY, do not scale with increasing dimensionality. Due to an overwhelming sets of cluster candidates in low dimensional projections they show exponentially increasing runtimes. INSCY did not even finish for the 55-dimensional within 48 hours. In contrast, approximative solutions like FIRES [KKRW05] scale slightly better, but as depicted in Figure 9.8(b) provide only very poor clustering quality. They miss relevant subspace clusters due to their poor approximation of density-based subspace clustering. As they use no steering logic, but only a single estimation phase and immediate output, their subspace clustering amounts to a heuristic guess. Our density-based subspace clustering approach steers the subspace cluster search directly to subspace clusters which contribute to the result set in a principled fashion, and therefore provides best quality.

As comparison baseline, we have depicted also projected clustering approaches. Since they require an input parameter on the number of clusters that should be detected, they are provided with the true hidden number of clusters in a data set. Thus, in these experiments they have the unfair advantage of having more knowledge about the data, which in real world applications is clearly not available. We can see that despite this additional information for the projected clustering competitors, our steering approach is capable of reliably selecting the best subspace clusters for the result set. Thus, the effort put into a more complex subspace clustering result which is capable of detecting overlapping subspace clusters, and which uses the robust density-based paradigm, pays off in terms of quality for our SSCS approach.

In terms of runtime, approaches like MINECLUS [YM03] achieve good results for low dimensional data due to their randomization of the search. However, they clearly show bad scalability. For the 105-dimensional data set MINELCUS did not finish within 48 hours. The simpler approach PROCLUS scales better with almost linear increase of runtime, but shows very bad clustering quality. Compared to PROCLUS, our approach achieves similar scalability of runtimes while providing constantly better quality results, not reached by any other subspace clustering algorithm. Please note that comparison with some algorithms that require the number of clusters as an input, like the projected clustering approach PROCLUS, does not reflect their performance in real application scenarios where this information is typically not available.

Our SSCS approach scales well with increasing dimensionality. It is less affected by the dimensionality as it jumps directly to most promising density-based subspace clusters excluding most low dimensional projections. Our steered processing and the

efficient gathering of information ensures an overall efficient computation. Although some algorithms have smaller runtime, our approach shows best performance with both efficient processing (cf. Fig. 9.8(a)) and high clustering quality results (cf. Fig. 9.8(b)).



(a) runtime for increasing database size

(b) runtime for increasing cluster dimensionality

Figure 9.9: Scalability w.r.t. size of the database and avg. cluster dimensionality

Database size Our next experiment in Figure 9.9(a) shows the runtime of the algorithms with respect to increasing database size. We use 20-dimensional databases for comparison with all density-based approaches, as INSCY does not run on most of our higher dimensional databases. For these data we observe best runtimes for both projected clustering approaches in the previous experiment. Our approach scales well with similar slope in runtime for increasing number of objects. Overall the results show that subspace clustering is more affected by dimensionality than by database size. Nevertheless, due to our efficient information gathering in different density levels we achieve far better runtimes than both density-based competitors.

Cluster Dimensionality Our last scalability experiment in Figure 9.9(b) shows the effect of increasing cluster dimensionality. INSCY is not depicted, as for this experiment we have used a high dimensional dataset with 65 dimensions. PROCLUS shows best runtimes due to its simple clustering model. For density-based approaches, our approach clearly outperforms FIRES, but we also outperform the projected MINECLUS approach that is heavily affected by the increasing cluster dimensionality. Our algorithm scales much better to higher dimensional clusters.

Overall, Figure 9.8 and Figure 9.9 illustrate the scalability of subspace clustering approaches especially with respect to the dimensionality of databases and hidden subspace clusters, which are crucial for efficient subspace clustering. We showed how well our approach scales both in terms of runtime and clustering quality. In the following experiments, we will give a detailed evaluation of density-based subspace clustering paradigm on real world data. As projected clustering approaches are not designed for detection of objects in multiple subspace clusters, and as they require the number of hidden clusters which is not given in real world data, PROCLUS and MINECLUS are not considered in the following experiments.

9.4.2 Evaluation on real world data

Our next experiment evaluates the performance on real world data, with results summarized in Table 9.2. We evaluate runtime (in seconds) and additionally measure cluster quality by F1 value and accuracy. In addition to the given databases Pendigits16, Face50 and Leaf50, we vary the dimensionality of the pendigits dataset from 16 to 48. For these three datasets we observe similar results as for scalability on the synthetic data. INSCY does not scale to more than the 16 dimensional pendigits dataset while the approximative FIRES approach has in most cases higher runtimes than our approach. We have highlighted in gray the lowest runtimes and highest cluster quality for each database. Our novel approach clearly outperforms existing density-based subspace clustering techniques. Although FIRES achieved lower runtime in two cases, it shows significantly lower quality on all databases.

	SSCS			FIRES			INSCY		
	runtime	F1 value	accuracy	runtime	F1 value	accuracy	runtime	F1 value	accuracy
Pendigits16	118	0,63	0,87	150	0,30	0,43	3937	0,51	0,57
Pendigits32	565	0,62	0,86	433	0,23	0,40	X	X	X
Pendigits48	1571	0,60	0,85	6562	0,27	0,50	X	X	X
Face50	982	0,21	0,59	2924	0,17	0,32	X	X	X
Leaf50	934	0,24	0,55	31	0,12	0,32	X	X	X

Table 9.2: Summary of real world data experiments

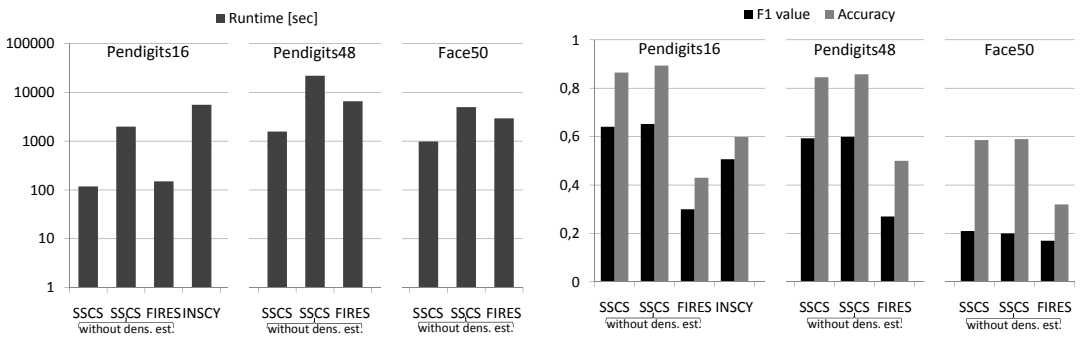
It should be highlighted that the results of the two density-based approaches INSCY and FIRES cannot compete with both efficiency and high quality results of our approach. We always achieve best clustering quality due to our efficient but also high quality information gathering. Steering subspace cluster search shows best performance detecting the hidden structures in the data with an efficient subspace processing.

Efficient Information Gathering We also evaluate the effect of our information gathering approach on the efficiency and quality of subspace clustering. In Figure 9.10 we compare two variants of our approach using different methods for information gathering. One variant (leftmost bar) with the density estimation layer presented in Section 9.3.2 and the other one (second bar) without this lowest layer.

In Figure 9.10(a) we show that with our density estimation step a significant efficiency improvement is achieved.

Although density estimation is a heuristic approach we see in Figure 9.10(b) that the clustering quality varies only marginally. Again, our SSCS keeps significantly higher quality than the approximative FIRES approach. Overall, we show both an efficient and high quality information gathering in our approach.

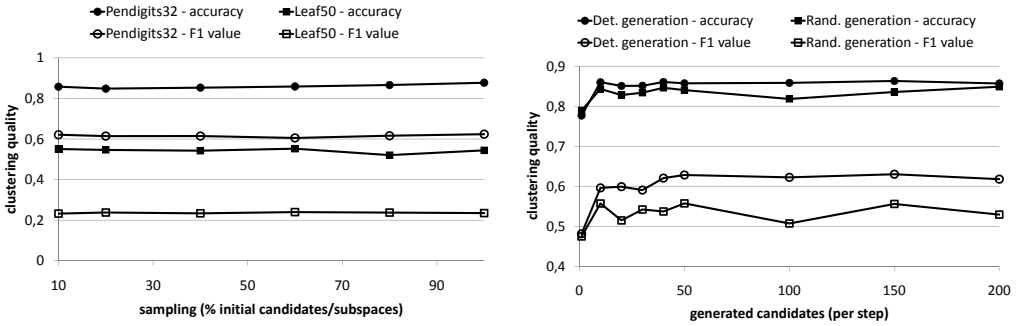
Parametrization of Steering Furthermore, we evaluate the main parameters of our steering method. We omit, that parameters controlling density estimation, discretization and other used techniques are of great importance for clustering quality but not evaluated in this chapter. We focus more on the novel steering as part of



(a) Runtime (b) Quality

Figure 9.10: Efficiency improvement

the jump processing schema to be evaluated. Based on two real world databases we show in Figure 9.11(a) that the initialization of our steering achieves a very robust clustering result. By sampling a set of subspace clusters as the initial knowledge base for our steering, we ensure efficient processing, but do not miss any meaningful clusters. A smaller number of samplings leads to a more efficient initialization phase of algorithm. The clustering quality is remarkably robust against the parameter *sampling*. Even for extremely small samples, cluster quality remains high due to continuous information gathering of the algorithm in the multi-jump process.



(a) Initialization (*sampling* variation) (b) Amount of extracted knowledge (*generatedClusters* variation)

Figure 9.11: Robustness w.r.t. parameter variation

Next, we evaluate the parametrization of the knowledge extracted by our multi-jumps. The amount of generated jump targets for each jump is important to ensure a high quality clustering result as shown in Figure 9.11(b). We varied the number of generated candidates from 1 to 200 and observe a drop of quality for very low numbers. However, by generating at least 30 candidates our approach shows reliably high clustering quality.

Overall, our experiments have shown best cluster quality results, achieved with our novel and efficient steering of the subspace cluster search.

9.5 Enhancements by steering the processing of subspace clusters

We introduce a steering method for efficient mining of density-based subspace clusters in high dimensional databases. Our novel steering incorporates efficient information gathering and feedback from progressively detected subspace clusters to tackle the major efficiency challenges of density-based subspace clustering. The exponential search space complexity and the high costs for exhaustive database scans are reduced resulting in an efficient processing which scales well with increasing number of dimensions. Thorough experiments demonstrate that our approach scales better than existing subspace and projected clustering algorithms showing efficient performance with high quality results.

Overall, the proposed steering approach is a fundamental technique in the development of efficient subspace clustering methods. Improving the efficiency of subspace clustering started by the development of bottom-up processing in various breadth-first methods [AGGR98, KKK04]. It has been enhanced by our multi-step architecture which could reduce the costly database scans to ensure efficient but also lossless subspace clustering (cf. Chapter 5). Further enhancements were achieved by our depth-first processing proposed in Chapter 6, introducing the general idea of in-process removal of redundant subspace clusters. And finally, the solutions proposed in Chapter 8 and Chapter 9 achieve to compute an efficient approximation of our NP-hard optimization for relevant subspace clusters. In general, in each of these techniques researchers have reduced the cluster computation to even fewer cluster candidates, down to the processing of only the most promising subspace clusters as proposed in this chapter.

Part III

Evaluation and exploration

Chapter 10

Systematic evaluation of clustering techniques in subspace projections

In the previous parts of this thesis the focus was on enhancement of subspace clustering techniques. We provided novel models for high quality subspace clustering results in Part I and developed novel processing schemes for efficient computation in Part II. However, like in all other publications in this area we focus our evaluation on a specific set of competitors to highlight our significant enhancements compared to the most relevant approaches. In addition to this focused evaluation, we propose a systematic evaluation study on a broad set of clustering techniques in the following. Please note, that for an objective and fair comparison of a broad set of approaches we included only one of our approaches in this evaluation study. However, the evaluation in this study is comparable to the previous chapters as all experiments are based on the same evaluation framework described in Chapter 11. This ensures not only comparability of the given results but also repeatability for future experiments.

As Clustering high dimensional data in *subspace clustering* or *projected clustering* is an emerging research field a systematic evaluation is very important for the research community. In the past decade, several clustering paradigms have been developed in parallel, without thorough evaluation and comparison on a common basis. Conclusive evaluation and comparison is challenged by three major issues. First, there is no ground truth that describes the "true" clusters in real world data. Second, a large variety of evaluation measures have been used that reflect different aspects of the clustering result. Finally, in typical publications authors have limited their analysis to their favored paradigm only, while paying other paradigms little or no attention.

In this chapter, we take a systematic approach to evaluate the major paradigms in a common framework. We study representative clustering algorithms to characterize the different aspects of each paradigm and give a detailed comparison of their properties. We provide a benchmark set of results on a large variety of real world and synthetic data sets. Using different evaluation measures, we broaden the scope of the experimental analysis and create a common baseline for future developments and comparable evaluations in the field. For repeatability, all implementations, data sets and evaluation measures are available on our website[1].

[1] http://dme.rwth-aachen.de/OpenSubspace/evaluation

10.1 Introduction to cluster detection in subspace projections

Knowledge discovery in databases provides database owners with new information about patterns in their data. Clustering is a traditional data mining task for automatic grouping of objects [HK01]. Cluster detection is based on similarity between objects, typically measured with respect to distance functions. In high dimensional spaces, effects attributed to the "curse of dimensionality" are known to break traditional clustering algorithms [BGRS99]. Meaningful clusters cannot be detected as distances are increasingly similar for growing dimensionality. To detect patterns obscured by irrelevant dimensions, global dimensionality reduction techniques such as principle components analysis (PCA) are not sufficient [Jol86]. By definition, they reduce the original high dimensional space to a single lower dimensional projection for all objects alike. In high dimensional spaces, however, dimensions might have locally varying relevance for different groups of objects. These cannot be detected by a global analysis of relevance. Recent research has introduced clustering in subspace projections, aiming at detecting locally relevant dimensions per cluster.

In several application scenarios like sensor networks, customer profiling, and bioinformatics high dimensional data is measured. Exemplary we highlight the requirement for cluster detection in gene expression analysis [CC00], as research on clustering in high dimensional data started with this application domain. High throughput experiments of gene expressions were available and opened questions like '*which genes have common functions and should be grouped*'. Databases consist of genes (objects) described by expression levels in different experimental conditions (attributes). High dimensional data occur as there are very many different experimental conditions to be analyzed. In general the problem can be abstracted to a huge number of objects with various attributes as depicted in Figure 10.1. Possible clusters in subspace projections are highlighted in gray. In many recent applications like sensor networks, objects are also described by very many attributes. As collecting and storing data is cheap, users tend to record everything without considering the relevance for their task. Clustering of such high dimensional data has become a general challenge for a broader range of data.

Recent research for clustering in high dimensional spaces has introduced a number of different approaches. They were named by the pioneers in this field *subspace clustering* [AGGR98] or *projected clustering* [AWY+99]. Both terms were used in parallel for development of further approaches. Their common goal is to detect the most relevant subspace projections for any object in the database. Any cluster is then associated with a set of relevant dimensions in which this pattern has been discovered. These techniques have been successfully applied in a number of scenarios. We illustrate possible clusters with relevant subspace projections for an example database in Figure 10.1. Cluster 3 represents a traditional cluster in full space, while clusters 1, 2 and 4 appear only in a subset of relevant dimensions. Please note, that for both objects and dimensions an arbitrary subset can become a subspace cluster. Projected clustering algorithms are restricted to disjoint sets of objects, while

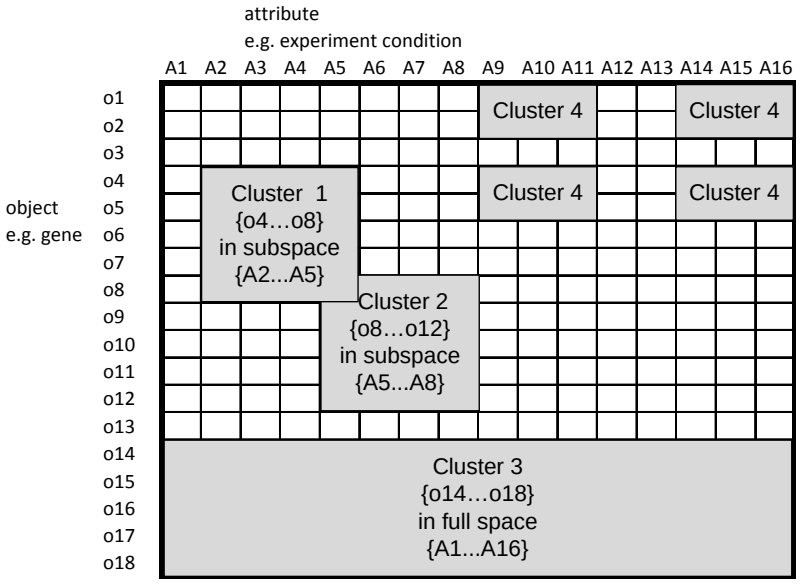


Figure 10.1: Example for subspace clustering

subspace clustering algorithms might report several clusters for the same object in different subspace projections. Motivated by the gene expression analysis, a gene may have several function represented by clusters with different relevant attributes (cf. object 8 in Fig. 10.1). For simplicity of presentation, we choose *subspace clustering* as the preferred term.

For evaluation and comparison, of subspace clustering algorithms in general, the last decade has seen several paradigms, characterized by their underlying cluster models and their parametrization of the resulting clustering. In this young field, however, we lack a common ground for evaluation as a whole. Three major problems persist. First, there is no ground truth that describes the "true" clusters in real world data. Second, a large variety of evaluation measures have been used that reflect different aspects of the clustering result. Finally, in typical publications authors have limited their analysis to their favored paradigm only, while paying other paradigms little or no attention. This implies several problems for the advancement of the field. Properties of the different paradigms are not yet well understood, as cross-comparisons are not available. The same is true for evaluation measures. They reflect different aspects which are not yet fully understood. It is therefore not possible to compare the results that have been reported in different papers. As a consequence, there is no common basis for research in the area which implies misleading conclusions from reported experiments, and hence possibly wrong assumptions about the underlying properties of algorithms.

In this work, we provide a systematic and thorough evaluation of subspace clustering paradigms. Results are analyzed using the measures that have been proposed by researchers in recent papers. We use a large collection of data sets, synthetic data with known hidden clusters and also publicly available real world data sets. Our work provides a meaningful characterization of the different paradigms and how these are reflected in the evaluation measures. We create a common ground on

which future research in the field can build. Our analysis uses our own open source framework described in the following chapter. This framework extends the popular open source WEKA platform that has been successful for full space data mining algorithms [WF05]. Our work is therefore easily repeatable and extensible for future algorithms. Implementations, data sets and measures are all available for anyone interested in comparing their own algorithms or the ones evaluated in this thesis.

This chapter is structured as follows: in Section 10.2 we review existing subspace clustering paradigms and point out their characteristics. Section 10.3 introduces the different measures used to evaluate existing algorithms in the literature. Our experimental evaluation in Section 10.4 gives a detailed analysis of the different paradigms under these measures. Finally, we conclude with discussion of our findings and highlight the contributions of our evaluation to future research in Section 10.5.

10.2 Clustering in subspace projections

Clustering in subspace projections aims at detecting groups of similar objects and a set of relevant dimensions for each object group. While there are two different names in the literature, *subspace clustering* [AGGR98] and *projective clustering* [AWY+99], we identify three major paradigms characterized by the underlying cluster definition and parametrization of the resulting clustering:

First, cell-based approaches search for sets of fixed or variable grid cells containing more than a certain threshold many objects. Subspaces are considered restrictions of a cell in a subset of the dimensions, while in the residual dimensions the cell spans the whole attribute domain. Cell-based approaches rely on counting objects in cells and with their discretization of data are similar to frequent itemset mining approaches.

Second, the density-based clustering paradigm defines clusters as dense regions separated by sparse regions. As density computation is based on the distances between objects, in subspace clustering one computes distances by taking only the relevant dimensions into account. Density-based approaches are thus dependent on the distance definition. They can be parametrized by specifying which objects should be grouped together due to their similarities w.r.t. distances.

Finally, clustering-oriented approaches do not give a cluster definition like the previous paradigms. In contrast, they define properties of the entire set of clusters, like the number of clusters, their average dimensionality or more statistically oriented properties. As they do not rely on counting or density they are more flexible in handling different types of data distributions. However, they do not allow parametrization of each cluster.

For each of the three main paradigms we evaluate seminal approaches which defined these paradigms and evaluate also the most recent representatives. It is clearly not possible to include in this study all algorithms from all of these paradigms. Also, it is beyond the scope of this work to include more specialized algorithms like correlation clustering [AY00] which transform the data space based on detected correlations, or application dependent approaches popular e.g. in bioinformatics [CC00]. Furthermore, we consider subspace clustering only on continuous valued attributes.

paradigm	approach	properties
cell-based	CLIQUE [AGGR98]	fixed threshold, fixed grid, pruning by monotonicity property
cell-based	DOC [PJAM02]	fixed result size, fixed threshold, variable hypercubes, randomized, partitioning
cell-based	MINECLUS [YM03]	enhances DOC by FP-tree structure [HPY00] resulting in more efficient mining
cell-based	SCHISM [SZ04]	enhances CLIQUE by variable threshold, using heuristics for approximative pruning
density-based	SUBCLU [KKK04]	fixed density threshold, pruning by monotonicity property
density-based	FIRES [KKRW05]	variable density threshold, using 1d histograms for approximative pruning
density-based	INSCY [AKMS08b]	variable density threshold, reducing result size by redundancy elimination
clustering-oriented	PROCLUS [AWY+99]	fixed result size, iteratively improving result like k-means [Mac67], partitioning
clustering-oriented	P3C [MSE06]	statistical tests, using EM [DLR77] clustering, pruning by monotonicity property
clustering-oriented	STATPC [MS08]	statistical tests, reducing result size by redundancy elimination, approximative

Table 10.1: Characteristics of three major paradigms

Subspace clustering of categorical attributes is a specialization of the frequent itemset mining task, and heterogeneous data is just a very recently upcoming topic as described in Chapter 7.

We consider an abstract high dimensional database with objects described by various attributes. As exemplified in Figure 10.1, a subspace projection is an arbitrary subset of attributes. Each cluster is described by a subset of objects (rows) and a subset of attributes (columns). Please note that in some approaches clusters in subspace projections may overlap in both objects and dimensions, as similarity between the objects is only evaluated with respect to the relevant dimensions.

To study different paradigms we use various evaluation measures that are described in more details in Section 10.3. Both efficiency in terms of runtime and also clustering quality in terms of different measures are analyzed. For our review on different subspace clustering paradigms we thus highlight both the effect of the cluster model on the quality, as well as the effect of algorithmic properties on the runtime. An overview of all paradigms, the used approaches and a summary of their important properties is given in Table 10.1.

Notations For consistent notations in the following sections we abstract from the individual definitions in the literature. Every cluster C in a subspace projection is defined by a set of objects O that is a subset of the database DB and a set of relevant dimensions S out of the set of all dimensions D.

A cluster C in a subspace projection S is

$$C = (O, S) \text{ with } O \subseteq DB, S \subseteq D$$

A clustering result is a set of found clusters in the respective subspace projections. A clustering result R of k clusters is a set of clusters

$$R = \{C_1, \ldots, C_k\}, C_i = (O_i, S_i) \text{ for } i = 1 \ldots k$$

We define several basic objective functions to describe the clustering result. The number of detected clusters is given by $numClusters(R) = k$. The average dimensionality of the clusters in the result is $avgDim(R) = \frac{1}{k} \cdot \sum_{i=1}^{k} |S_i|$. For ease of presentation and w.l.o.g. we assume each dimension has the same domain, thus, $domain(DB) = [0..v]^{|D|}$.

10.2.1 Cell-based approaches

First, we consider the cluster definition and its parameterization. Cell-based clustering is based on a cell approximation of the data space. Cells of width w are used to describe clusters. For all cell-based approaches, a cluster result R consists of a set of cells; each of them containing more than a threshold τ many objects ($|O_i| \geq \tau$ for $i = 1 \ldots k$). These cells describe the objects of the clusters either by a hypercube of variable width w [PJAM02, YM03] or by a fixed grid of ξ cells per dimension [AGGR98, SZ04]. Fixed grids can be seen as discretization of the data space in pre-processing. In contrast, variable hypercubes are arbitrarily positioned to delimit a region with many objects.

Definition 37. Cell-Based Subspace Cluster.

A cell-based subspace cluster (O, S) is defined w.r.t. minimum number of objects τ in cells CS of w width specified by intervals I_i per dimension $\forall i \in S$. Each interval is part of the common domain $I_i = [l_i \ldots u_i] \subseteq [0 \ldots v]$ with lower and upper bound l_i and u_i. For all non-relevant dimensions $\forall j \in D \setminus S$ the interval is the full domain $I_j = [0 \ldots v]$ i.e. the cell is not restricted in these dimensions. The clustered objects $O = \{o \mid o \in DB \cap CS\}$ fulfill $|O| \geq \tau$

The first approach for cell-based clustering was introduced by CLIQUE [AGGR98]. CLIQUE defines a cluster as a connection of grid cells with each more than τ many objects. Grid cells are defined by a fixed grid splitting each dimension in ξ equal width cells. Arbitrary dimensional cells are formed by simple intersection of the 1d cells. First enhancements of CLIQUE adapted the grid to a variable width of cells [NGC01]. More recent approaches like DOC use flexible hypercubes of width w [PJAM02]. In MINECLUS such hypercube approaches are supported by FP-trees, known from frequent itemset mining to achieve better runtimes [HPY00, YM03]. SCHISM, improves the cluster definition by variable thresholds $\tau(|S|)$ adapting to the subspace dimensionality $|S|$ [SZ04].

Second, we consider efficiency. As subspace clustering searches for clusters in arbitrary subspaces, naive search is exponential in the number of dimensions. CLIQUE proposes a pruning criterion for efficient subspace clustering based on a monotonicity property. A similar monotonicity property was introduced in the apriori algorithm [AS94] for efficient frequent itemset mining and has been adapted to subspace clustering [AGGR98]. Monotonicity is used by most subspace clustering algorithms, and states that each subspace cluster (O, S) appears in each lower dimensional projection T, i.e. $\forall T \subset S \ : \ (O, T)$ is also a subspace cluster. The negation of this monotonicity can then be used for pruning in a bottom-up algorithm on the subspace lattice: If a set of objects O does not form a cluster in subspace T then all higher dimensional projects $S \supset T$ do not form a cluster either.

It is important to highlight two major characteristics of this pioneer work in clustering subspace projections: First, a monotonicity property is the most common way in pruning subspaces. It has been applied also in other paradigms for efficient computations of subspace clusters. And second, the cell-based processing of the data space has been used in several other techniques to efficiently compute regions with at least a minimal amount of objects.

Although there are some differences, cell-based approaches share a main common property. They all count the number of objects inside a cell to determine if this cell is part of a subspace cluster or not. This counting of objects is comparable to frequency counting in frequent itemset mining. Subspace clusters are sets of frequently occurring attribute value combinations in the discretized space. One abstracts from the original data distribution of continuous valued attributes and only takes the discretized (in or outside the cell) information into account. On the one side, this makes the computation more efficient, however, on the other side discretization may result in loss of information and possibly less accurate clustering results. Furthermore, quality of the result is highly dependent on cell properties like width and positioning.

Simple counting has further advantages as it is easy to parametrize. Giving a threshold for the number of objects in a cluster is very intuitive. However, as this is a property of a single cluster one has only little control on the overall clustering result. For example, the mentioned monotonicity of CLIQUE induces that for each detected subspace cluster all lower dimensional projections will also be clusters. This might result in a tremendously large clustering result R where $numClusters(R) \gg |DB|$ is possible.

10.2.2 Density-based approaches

Density-based approaches are based on the clustering paradigm proposed in DBSCAN [EKSX96]. They compute the density of each object by counting the number of objects in its ε-neighborhood without prior discretization. A cluster with respect to the density-based paradigm is defined as a set of dense objects having more than $minPoints$ many objects in their ε-neighborhood. Arbitrarily shaped clusters are formed by a chain of dense objects lying within ε distance of each other. To determine the neighborhood for each object, a distance function is used (typically Euclidean distance). By changing the underlying distance function and the ε parameter, one can

specify the range of similar objects to be grouped in one cluster. This parametrization of the similarity gives the approaches in this paradigm high flexibility, but requires knowledge about suitable choices for the data, often not available in unsupervised learning.

Definition 38. Density-Based Subspace Cluster.

A density-based subspace cluster (O, S) *is defined w.r.t. a density threshold* $minPoints$ *and* ε*-neighborhood* $N_\varepsilon(q) = \{p \in DB \mid dist^S(p, q) \leq \varepsilon\}$*, where* $dist^S$ *denotes a distance function restricted to the relevant dimensions* S*:*

All objects are dense: $\forall o \in O: \ |N_\varepsilon(o)| \geq minPoints$.
All objects are connected:
$\forall o, p \in O: \exists\ q_1, \ldots, q_m \in O: q_1 = o \ \wedge q_m = p \ \wedge \forall i \in \{2, \ldots, m\}\ q_i \in N_\varepsilon(q_{i-1})$.
The cluster is maximal: $\forall o, p \in DB:$ *o, p connected* $\Rightarrow (o \in O \Leftrightarrow p \in O)$

The first approach in this area was SUBCLU [KKK04], an extension of the DBSCAN algorithm to subspace clustering, by restricting the density computation to only the relevant dimensions. Using a monotonicity property, SUBCLU reduces the search space by pruning higher dimensional projections like CLIQUE. In contrast to cell-based approaches, the density-based paradigm uses the original data and requires expensive database scans for each ε-neighborhood computation. This results in an inefficient computation. A more efficient, however, approximative solution is proposed by FIRES [KKRW05]. Instead of going through the subspaces bottom up, FIRES uses 1d histogram information to jump directly to interesting subspace regions. A non-approximative extension of SUBCLU is our INSCY approach (cf. Chapter 6), which eliminates redundant low dimensional clusters, detected already in higher dimensional projections. In contrast to breadth-first approaches, INSCY processes subspaces recursively and prunes low dimensional redundant subspace clusters. Thus, it achieves an efficient computation of density-based subspace clusters.

The overall quality of density-based subspace clusters is dependent on the similarity specification. Similar to the dependency of cell-based approaches to their grid properties, finding meaningful parameter settings for the neighborhood range ε is a challenging task. FIRES uses a heuristic to adapt its $\varepsilon(|S|)$ to the subspace dimensionality $|S|$, however, it still has to initialize $\varepsilon(1)$. Similarly, INSCY uses a normalization for the density threshold $minPoints(|S|)$ for arbitrary subspaces as proposed in Chapter 2, keeping the ε parameter fixed. Both enhancements allow flexible parametrization, but, not totally eliminate the challenging task of finding an adequate similarity for arbitrary subspaces. Furthermore, like for cell-based approaches, individual cluster properties give almost no control over the final clustering result.

10.2.3 Clustering-oriented approaches

In contrast to the previous paradigms, clustering-oriented approaches focus on the clustering result R by directly specifying objective functions like the number of clusters to be detected or the average dimensionality of the clusters as in PROCLUS

[AWY+99], the first approach for this paradigm. PROCLUS partitions the data into k clusters with average dimensionality l, extending K-means [Mac67]. Instead of a cluster definition, clustering oriented approaches define properties of the set of resulting clusters. Each object is assigned to the cluster it fits best. More statistically oriented, P3C uses χ^2 test and the expectation maximization algorithm to find a more sophisticated partitioning [MSE06, DLR77]. Defining a statistically significant density, STATPC aims at choosing the best non-redundant clustering [MS08]. Although it defines cluster properties, it aims at an overall optimization of the clustering result R.

Definition 39. Clustering Oriented Results.

A clustering oriented result w.r.t. objective functions $f(R)$, which is based on the entire clustering result R and an optimal value parameter $optF$ (e.g. $numClusters(R) = k$ and $avgDim(R) = l$ in PROCLUS) is a result set R with: $f(R) = optF$.

The most important property for clustering-oriented approaches is their global optimization of the clustering. Thus, the occurrence of a cluster depends on the residual clusters in the result. Based on this idea, these approaches are parametrized by specifying objective functions for the resulting set of clusters. Some further constraints about the clusters like in STATPC are possible, but, the global optimization of the result is still the major goal.

Clustering-oriented approaches directly control the resulting clusters, e.g. the number of clusters. Other paradigms do not control such properties as they report every cluster that fulfills their cluster definition. Both cell-based and density-based paradigms provide a cluster definition; every set of objects O and set of dimensions S fulfilling this definition is reported as subspace cluster (O, S). There is no explicit optimization process to select clusters as all clusters are output that fulfill the given cluster definition. On the other side, clustering oriented approaches do not influence the individual clusters to be detected. They provide a general optimization criterion and do not specify a definition for a single cluster. For example, keeping the number of clusters fixed and partitioning the data, optimizes the overall coverage of the clustering like in PROCLUS or P3C, but, includes noise into the clusters. As these approaches optimize the overall clustering they try to assign each object to a cluster, resulting in clusters containing highly dissimilar objects (noise). Both approaches are aware of such effects and use outlier detection mechanisms to remove noise out of the detected clusters. As these mechanisms tackle noise after the optimization process, clustering quality is still affected by noisy data.

10.3 Evaluation measures

In this section we describe the measures used in our evaluation of the different subspace clustering algorithms. While the efficiency can easily be measured in terms of *runtime*, the quality is more difficult to determine. One problem is that there is usually no ground truth to which we can compare the clustering result $R = \{C_1, \ldots, C_k\}$. For the classification task in contrast one can easily use labeled data as ground truth

and compare these labels with the predicted labels of any classifier to obtain a reproducible quality measure. For clustering two possibilities to determine the ground truth are used. On synthetic data the "true" clustering is known a priori and hence the ground truth is given. We refer to these "true" clusters as the *hidden* clusters $H = \{H_1, \ldots, H_m\}$ in contrast to the *found* clustering result $R = \{C_1, \ldots, C_k\}$. For each $H_i = (O_i, S_i) \in H$ we can use information about the grouped objects O_i and the relevant dimensions S_i of the cluster, known by the data generator.
On real world data this information is not given. Therefore the idea is to use labeled data with the assumption that the natural grouping of the objects is somehow reflected by the class labels. All objects O_i with the same class label i are grouped together to one cluster $H_i = (O_i, S_i)$. Disadvantageous for this method is that the relevant dimensions of the cluster H_i cannot be deduced by the labels. However, assuming all dimensions to be relevant for each cluster (i.e. $S_i = D$) one can define for real world data also the hidden clusters $H = \{H_1, \ldots, H_m\}$.

We categorize the measures in two types depending on the required information about the hidden clusters H. The measures in the first category use information on which objects should be grouped together, and thus, form a cluster. Consequently only the information O_i out of each hidden cluster H_i is regarded. The second category of measures is based on the full information about the hidden subspace clusters. The objects O_i and the relevant dimensions S_i of each hidden cluster must be given to calculate these measures. The application of these measures to real world data is somehow restricted as typically the relevant dimensions are not available as ground truth for the hidden clusters. We constantly set the relevant dimensions for such data to D. By this procedure, full-space clusters are preferred over potentially more meaningful subspace clusters. Nonetheless one can use these measures to judge the grouping of the data objects based on the information O_i.

10.3.1 Object-based measures

Entropy [AKMS07a, SZ04]: The first approach is to measure the homogeneity of the found clusters with respect to the hidden clusters. A found cluster should mainly contain objects from one hidden cluster. The merging (splitting) of several hidden clusters to one (different) found cluster (clusters) is deemed to be a low quality cluster. The homogeneity can be measured by calculating the entropy, an information theoretic measure. Based on the relative number $p(H_i|C) = \frac{|O_{H_i} \cap O|}{|O|}$ of objects from the hidden cluster $H_i = (O_{H_i}, S_{H_i})$ that are contained in the found cluster $C = (O, S)$, the entropy of C is defined as:

$$E(C) = -\sum_{i=1}^{m} p(H_i|C) \cdot \log(p(H_i|C))$$

The overall quality of the clustering is obtained as the average over all clusters $C_j \in R$ weighted by the number of objects per cluster. By normalizing with the maximal entropy $log(m)$ for m hidden clusters and taking the inverse, the range is

between 0 (low quality) and 1 (perfect):

$$1 - \frac{\sum_{j=1}^{k} |C_j| \cdot E(C_j)}{\log(m) \sum_{j=1}^{k} |C_j|}$$

Hence the entropy measures the purity of the found clusters with respect to the hidden clusters.

F1 [AKMS08b, MAK+09]: The next measure evaluates how well the hidden clusters are represented. The found clusters which represent a hidden cluster should cover many objects of the hidden cluster but few objects from other clusters. This idea can be formulated with the terms recall and precision. Let $mapped(H) = \{C_1, \ldots, C_l\}$ (later described) be the found clusters that represent the hidden cluster H. Let O_H be the objects of the cluster H and $O_{m(H)}$ the union of all objects from the clusters in $mapped(H)$. Recall and precision are formalized by:

$$recall(H) = \frac{|O_H \cap O_{m(H)}|}{|O_H|} \quad precision(H) = \frac{|O_H \cap O_{m(H)}|}{|O_{m(H)}|}$$

A high recall corresponds to a high coverage of objects from H, while a high precision denotes a low coverage of objects from other clusters. The harmonic mean of precision and recall is the F1-measure where a high F1-value corresponds to a good cluster quality. The average over the F1-values for all hidden clusters $\{H_1, \ldots, H_m\}$ is the F1-value of the clustering:

$$\frac{1}{m} \sum_{j=1}^{m} \frac{2 \cdot recall(H_j) \cdot precision(H_j)}{recall(H_j) + precision(H_j)}$$

Different mappings of the found clusters to the hidden clusters are used in the literature. In our evaluation we perform a widely used mapping, where each found cluster is mapped to the hidden cluster which is covered to the most part by this found cluster. Formally, $C_i \in mapped(H)$ iff

$$\frac{|O_i \cap O_H|}{|O_H|} \geq \frac{|O_i \cap O_{H_j}|}{|O_{H_j}|} \quad \forall j \in \{1, \ldots, m\}$$

Please note, that for this widely used mapping counterexamples exist, where F1 provides perfect quality values although the observed clustering quality is bad. One such drawback of F1 is observed for a bad clustering where each object is detected in a single cluster. Showing both perfect recall and precision, this bad result has the highest possible F1 measure. Even though such theoretical drawbacks are known, in practice we do not observe such a bias of the F1 measure while other measures show clearly biased quality assessment as highlighted in our experiments. More enhanced mappings to solve the theoretical drawbacks of the F1 measure are still open research questions.

Accuracy [MAK+09, BZ07]: Another measure uses the accuracy of classification specified by $\frac{|\text{correctly predicted objects}|}{|\text{all objects}|}$ to judge the clustering quality. The idea is to predict the hidden cluster of an object on the basis of the detected patterns (i.e. the found clusters). The higher the accuracy the better is the generalization of the data set by the found clusters. The found clusters are a good description of the hidden clusters and hence the clustering quality is high. For quality measurement in recent publications, a decision tree classifier is build and evaluated (C4.5 with 10-fold cross validation) [BZ07]. To train the classifier the 'extracted dimensions' out of the clustering are used. Each object o is therefore represented as a bitvector of length k if we found k clusters $\{C_1, \ldots, C_k\}$. The position j in the bitvector equals 1 if $o \in C_j$, otherwise 0.
Please note that typically the classification accuracy based on the original dimensions of the objects instead of the extracted dimensions is higher because the bitvectors contain only knowledge that is generated by an unsupervised learning task.

10.3.2 Object- and subspace-based measures

Up to now no measure accounts for the relevant dimensions of a subspace cluster. However for synthetic data sets this information is available. The basic idea used in the next two measures is to consider the subspaces as follows: Instead of regarding the original database objects for the evaluation, each object is partitioned into subobjects annotated with a dimension. In a d-dimensional database the object o is partitioned in d different objects $o_1, \ldots, o_d$. A subspace cluster is henceforward not a subset of objects and a subset of dimensions but only a subset of these new subobjects. As a consequence two subspace clusters that share original objects but have disjoint relevant dimensions do not share subobjects. On the basis of this new representation further measures can be described.

RNIA [PM06]: A first approach is the relative nonintersecting area (RNIA) which measures to which extent the hidden subobjects are covered by the found subobjects. For a good clustering it is desirable to cover all and only the hidden subobjects. Formally one determines the subobjects which are in a hidden or found cluster (union U of subobjects) and subtracts the number of subobjects which are both in a hidden and found cluster (intersection I of subobjects). The more equal I and U the more equal are the found and hidden clustering and hence the better the clustering quality. To normalize the measure the term $RNIA = (U - I)/U$ is calculated. In the evaluation, we plot the value $1.0 - RNIA$ so that the maximal value 1 corresponds to the best clustering.

CE [PM06]: An extended version of RNIA is the clustering error (CE). One problem for the RNIA measure is that one cannot distinguish if several found clusters cover a hidden cluster or exactly one found cluster matches the hidden cluster. The RNIA-value is in both cases the same even though the second case is usually preferable. Therefore the CE measure maps each found cluster to at most one hidden cluster and also each hidden cluster to at most one found cluster. For each such mapping of two clusters the intersection of the subobjects is determined. Summing

up the individual values gives us a value $\bar{I}$. Substituting the value I by $\bar{I}$ in the RNIA formula results in the CE-value. In this way one penalizes clustering results which split up a cluster in several smaller ones (with respect to objects or dimensions).

Both measures, CE and RNIA, were implemented in versions that can handle also nondisjoint clusterings as described in [PM06].

We perform thorough evaluation on all measures of the two categories because there is no best solution so far. Each measure has its advantages and disadvantages depicted in the experimental section. Furthermore, for some properties like redundancy-removal or the consideration of relevant subspaces in real world data there exist no measure yet.

10.4 Experiments

In our thorough evaluation, we focus on the general properties of the clustering paradigms. For comparability, we implemented all algorithms in our evaluation framework. By extending the popular WEKA framework we base our work on a widely used data input format for repeatable and expandable experiments [WF05]. We used original implementations provided by the authors and best-effort re-implementations based on the original papers. The authors of SUBCLU, FIRES and MINECLUS provided us with original implementations, which we only adapted to our framework. For all other approaches, we re-implemented the approaches in our framework as no publicly available implementations exist so far. We ensure comparable evaluations and repeatability of experiments, as we deploy all implemented algorithms and parameter settings on our website[2].

With INSCY, we include one of our own subspace clustering algorithms as part of the density-based clustering paradigm without compromising objectivity and independence in evaluation. However, as it is our own approach we have more knowledge about parameterizing it. Nonetheless, our analysis is independent, as we evaluated a broad range of parameter settings for each algorithm to find the best parameters on each data set. We thus claim to provide an independent evaluation for clustering in subspaces of high dimensional data.

For a fair evaluation we ran massive experiments with various parameter settings for each algorithm. For the sake of space, we show an aggregated view of the results. Due to the enormous amount of experiment runs (23 data sets $\times$ 10 algorithms $\times$ on average 100 parameter settings per algorithm), we had to restrict the runtime for each run to 30 minutes. Based on preliminary experiments we observed runtimes of several days for some algorithms, which are clearly impractical. Experiments were run on a compute cluster with compute nodes equipped with four quad core Opteron 2.3 GHz CPUs running Windows 2008 Server. Java 32-bit runtime environment has been limited to using 1.5GB of RAM for each experiment. For repeatability, please refer to our website for an exhaustive list of parameter settings for each experiment.

[2] http://dme.rwth-aachen.de/OpenSubspace/evaluation

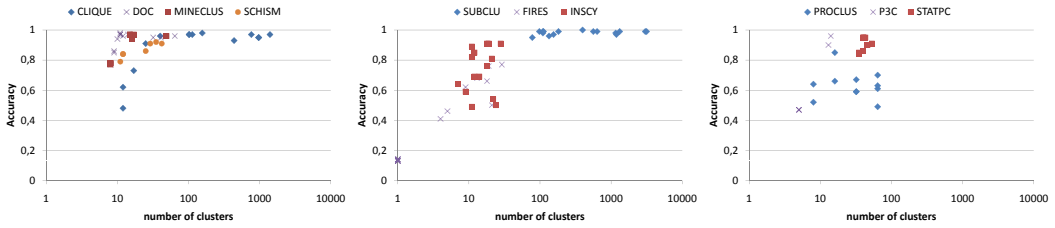
10.4.1 Clustering output (quality)

For a meaningful clustering result different aspects are important. First of all, clustering should detect only a small set of clusters, far less than the number of objects. Second, the detected clusters should represent the hidden structures in the data as closely as possible. In the following experiments we will give more insights in these two properties for each paradigm. We generated data with 10 hidden subspace clusters with a dimensionality of 50%, 60% and 80% of the five dimensional data space. In Figures 10.2, 10.3, 10.4, 10.5 and 10.6 we show dependencies between the number of found clusters and different quality measures as introduced in Section 10.3. Out of various parameter settings we picked the best five results for each of the presented measures. Thus, we realize comparability with publications using only one of these measures. In addition, we extend comparison to a broader set of measures to achieve objective results.

For ease of illustration we depict each paradigm in a separate figure but on identical scales. First, let us consider the number of clusters on the x-axis. For each paradigm we observe different characteristics: The basic approaches of cell-based and density-based clustering, CLIQUE and SUBCLU tend to produce a huge amount of clusters (> 1000) to achieve good results as shown in Figure 10.2. It is a common property for the more recent representatives of the two paradigms to achieve good quality with fewer clusters. In contrast, clustering oriented approaches in general produce only very few clusters. Their clustering optimization leads to high clustering quality already with 10 to 100 clusters.

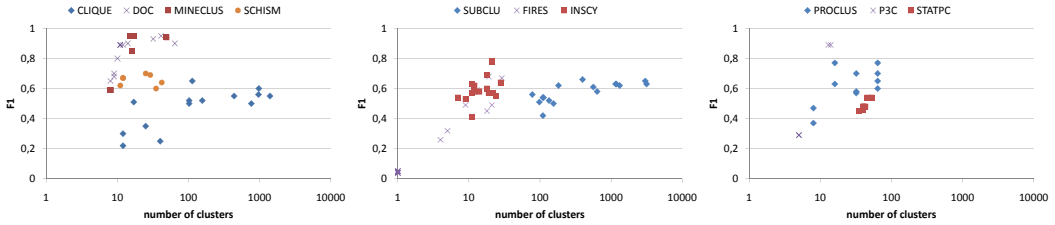
The distribution in number of clusters can be observed throughout the following figures with different quality measures on the y-axis. In Figure 10.2 we observe that Accuracy shows increasing quality with more and more detected clusters. The measure is correlated with the number of clusters. In contrast, for the F1 measure we do not observe such a correlation (cf. Figure 10.3). Although the formal definition of F1 would provide perfect quality for large result sizes (e.g. a clustering where each object is detected in a separate cluster), this drawback is not observed in empirical results. Most clustering algorithms provide rather large clusters, showing lower quality due to low purity inside each of these clusters. In addition, in Figures 10.5 and 10.6 the RNIA and CE measure show a peak around 10 clusters. This is exactly the number of clusters hidden in the synthetic data set. For more clusters both measures decrease as hidden structures are split up into more clusters. CLIQUE and SUBCLU have only low clustering quality w.r.t. RNIA and CE.

As a next aspect we want to analyze the cluster distribution w.r.t. average dimensionality of the detected clusters. In Figure 10.7 we show the result distribution for all parameter settings of the cell-based clustering approach DOC w.r.t. CE vs. average dimensionality. Please keep in mind that the CE measure takes not only objects but also the relevant dimensions into account. Thus, we see best results for three to four dimensional subspaces as we have three and four dimensional clusters hidden in the data. We show only the CE measure and the cell-based paradigm, but for RNIA and all other approaches a similar distribution has been observed. However, in contrast to other measures like F1 and Accuracy the CE measure highlights clustering quality



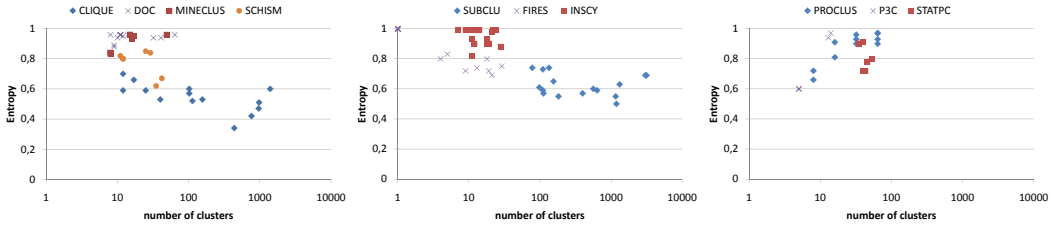
(a) cell-based (b) density-based (c) clustering-oriented

Figure 10.2: **Accuracy measure** vs. **number of clusters**



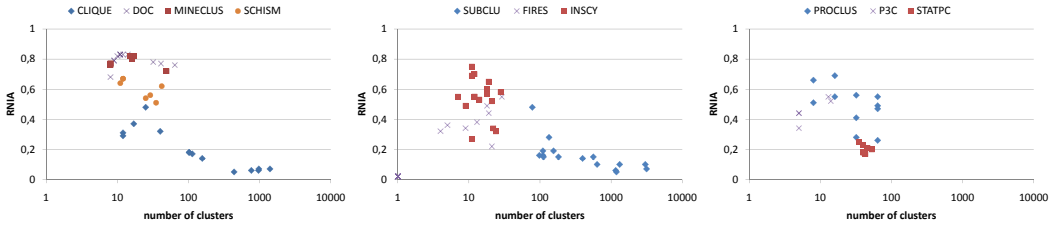
(a) cell-based (b) density-based (c) clustering-oriented

Figure 10.3: **F1 measure** vs. **number of clusters**



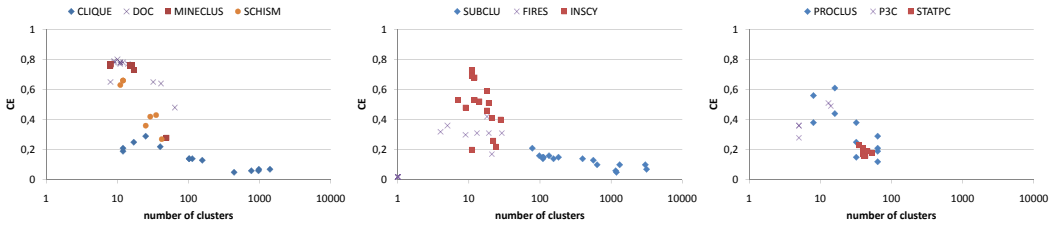
(a) cell-based (b) density-based (c) clustering-oriented

Figure 10.4: **Entropy measure** vs. **number of clusters**



(a) cell-based (b) density-based (c) clustering-oriented

Figure 10.5: **RNIA measure** vs. **number of clusters**



(a) cell-based (b) density-based (c) clustering-oriented

Figure 10.6: **CE measure** vs. **number of clusters**

if the right objects are detected as clusters in the right subspaces.

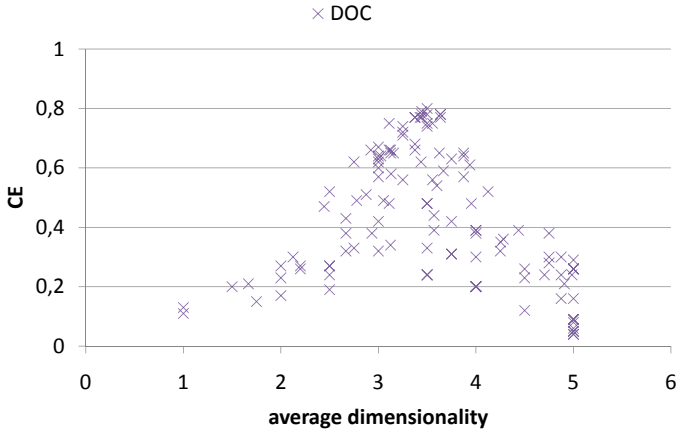


Figure 10.7: **CE measure** vs. **average dimensionality**

At last, we want to compare some measures among each other and point out advantages or disadvantages. First, we analyze the entropy and F1 measure. While the entropy is based on the found clusters, the F1 measure focuses on the hidden clusters. The problem by focusing on the found clusters is that in some sense the found clusters are regarded as the "true" clusters. This could lead to misleading results. For exemplification let us consider the case of an algorithm which detects only one out of several hidden clusters. If this found cluster is perfectly pure (i.e. only objects from one hidden cluster are contained), entropy reports optimal quality even though several other hidden clusters are not identified. Generally, entropy is biased towards high dimensional and therefore usually small and pure clusters (cf. Fig. 10.4(b) INSCY).

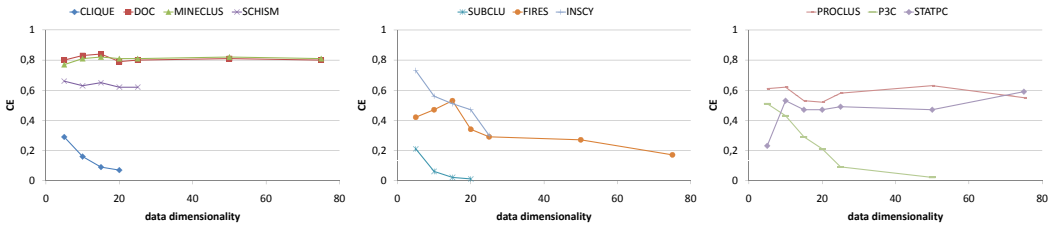
On the other hand, the F1 measure evaluates also detection of the hidden clusters. Not only the purity of the found clusters is important (resulting in a high precision), but also the detection of all hidden clusters (to get a high recall). Because both aspects are considered, the F1 measure is usually lower than the entropy measure (cf. Fig. 10.3 vs. 10.4), but also more meaningful.

The most important advantage of the last two measures RNIA and CE in comparison to Entropy, F1 and Accuracy was already mentioned in Section 10.3 and illustrated in Figure 10.7: Both measures consider the relevant dimensions of the clusters so that more meaningful quality values for synthetic data can result. Now we compare these two measure among each other. Looking at Figure 10.5 and 10.6 we see that the quality for RNIA is always larger than the CE quality. This is due to the fact that the RNIA measure do not penalize clusterings that distribute the found clusters over several hidden clusters. The difference in both measures however becomes smaller (e.g. for SUBCLU in Fig. 10.5(b) and 10.6(b)) if the result size if very large. Because of the large number of clusters the probability that a found cluster is mapped to a nearly identical hidden cluster increases and hence the difference in the intersection I in RNIA and $\bar{I}$ in CE (cf. Section 10.3.2) is small.
Another property of both measures is that for a huge result size the measures report

poor clustering quality. This is in contrast to the accuracy measure which has usually improved quality w.r.t. the result size.

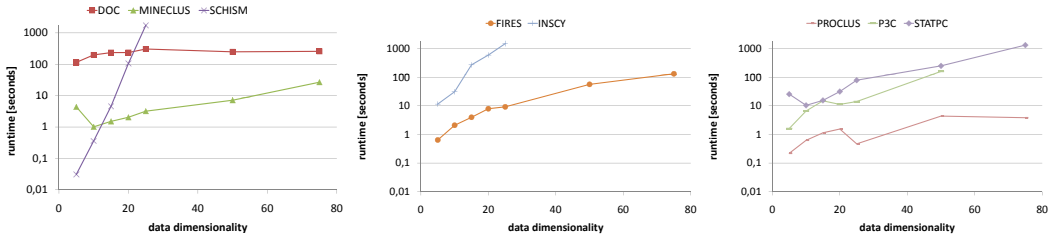
Because of the advantages of the CE measure on synthetic data, i.e. the consideration of the relevant subspaces, the penalization of high result sizes and the improvement over RNIA, we use this measure in the following experiments.

10.4.2 Scalability (efficiency and quality)



(a) cell-based (b) density-based (c) clustering-oriented

Figure 10.8: Scalability: **CE measure** vs. **database dimensionality**



(a) cell-based (b) density-based (c) clustering-oriented

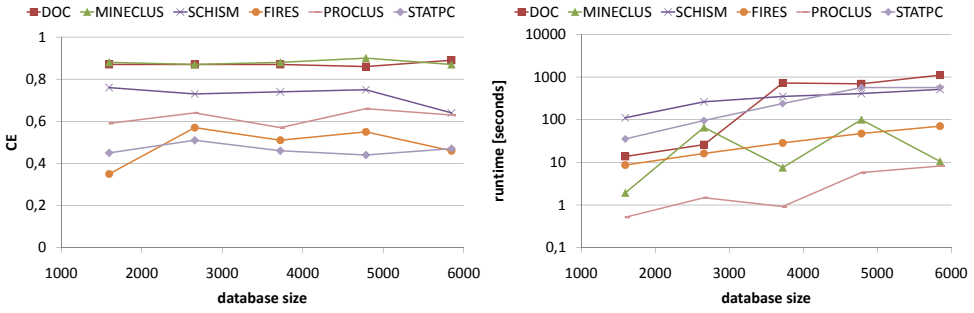
Figure 10.9: Scalability: **runtime** vs. **database dimensionality**

For scalability w.r.t. dimensionality of the database, we use synthetic data sets with 5-75 dimensional data. We generate data of different dimensionalities and hide 10 subspace clusters with a dimensionality of 50%, 60% and 80% of the data dimensionality. In Figure 10.8 we show clustering accuracy based on CE measure. We check quality both on object groups and detected subspaces as we know this for the synthetic data sets. Please note that for some algorithms due to extremely high runtimes we could not get meaningful results. In each paradigm at least one such algorithm is shown in Figure 10.8 where the quality measure decreases down to zero as there is not enough time to detect the hidden clusters. Preliminary experiments have shown that some algorithms result in runtimes of several days. The authors of SUBCLU published runtimes of up to six days [KKK04]. Such extreme effects were observed especially for high dimensional data ($|D| \geq 25$) for several algorithms (CLIQUE, SUBCLU, INSCY, P3C). It is clearly impractical to evaluate algorithms with such high runtimes on high dimensionalities.

In general, cell-based approaches, except CLIQUE, show best results ($CE : 0.6 - 0.8$) for all dimensionalities. For clustering oriented approaches we observe medium to high quality ($CE : 0.4 - 0.7$). The density-based paradigm shows the worst quality

measures with good results (CE : $0.4-0.7$) only for low dimensional data ($|D| < 20$). Due to high runtimes all density-based approaches (and also CLIQUE and P3C in the other paradigms) are not scalable to higher dimensional data as they cannot detect the clusters anymore ($CE < 0.3$).

Efficiency is a major challenge in clustering of high dimensional data. In Figure 10.9 we depict the runtimes for the algorithms which detected the hidden clusters within the limit of 30 minutes. The runtimes of MINECLUS and PROCLUS show best runtimes for different dimensionalities. All other approaches show either significant increase of runtime for higher dimensional data sets or constantly high runtimes even for low dimensional data. Depending on the underlying clustering model, algorithms always have to tackle the trade-off between quality and runtime. Typically, high quality results have to be paid with high runtime. For cell-based approaches, DOC requires very many runs to achieve its very good clustering quality on our data sets, while SCHISM falls prey to the dramatically increasing number of fixed grid cells in high dimensional spaces. The density-based approaches in general do not scale to high dimensional data due to their expensive database scans for neighborhood density computation. Both their runtimes increase and thus high quality results are not achieved within the limited time in our evaluation. For clustering-oriented approaches statistic evaluations in both P3C and STATPC do not scale w.r.t. dimensionality.



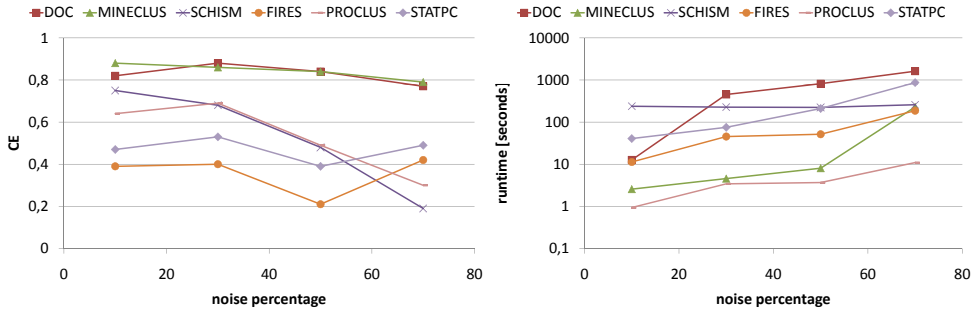
(a) CE measure (b) time

Figure 10.10: Scalability: **CE measure and runtime** vs. **database size**

Scalability w.r.t. database size is an important challenge especially for very large databases. In Figure 10.10(a) we show CE measure, while Figure 10.10(b) depicts the runtimes. We generated synthetic data by varying the number of objects per cluster, while keeping dimensionality fixed to 20d. We include only those algorithms which showed good quality results in this dimensionality in the previous experiments.

The cell-based paradigm outperforms the other paradigms also w.r.t. larger database sizes. It only slightly varies in clustering quality. Density-based and clustering oriented approaches show higher variance and overall significantly lower clustering quality. Considering runtime, the time to detect the clusters increases with the number of given objects for all approaches.

As clustering aims to detect clusters, it is challenging to cope with noisy data where some objects do not fit to any cluster. For the following experiment, we increase the percentage of noise objects in the database from 10% noise up to 70% noise, using the 20d database from the first scalability experiment. Figure 10.11



(a) CE measure (b) time

Figure 10.11: Scalability: **CE measure and runtime** vs. **noise percentage**

depicts quality and runtime results. As for database scalability we skip approaches that did not cluster in the given time. For clustering quality we see a significant decrease for almost all paradigms. Especially the clustering oriented approaches like PROCLUS are highly affected by the increase of noise. In terms of runtime we observe dramatically increasing runtimes for the cell-based approaches. Due to the increased amount of noise, more cells tend to have enough objects to form a cluster, but still DOC and MINCLUS achieve best quality results. For the remaining algorithms, this leads to both a decrease in quality as clusters which mainly contain noise are detected, and an increase in runtime as overall more clusters are detected. In general, a partitioning of the data is not always meaningful as noise should not be assigned to any cluster.

10.4.3 Real world data

We use benchmark real world data from the UCI archive [AN07]. These data sets, all of them or a subset, have been used for performance evaluations in recent publications [SZ04, MS08]. In addition, we use 17-dimensional features as extracted in our previous work from the sequence data in [KWX+06]. In contrast to some of the data sets used in other publications, all of the data sets are publicly accessible. For repeatability we provide them along with the synthetic data sets on our website. For each of the data sets we have optimized parameters for each algorithm based on the resulting F1 and Accuracy measure. As the real world data sets are typically used for classification tasks, they all have class labels. However, there is no information about the relevance of dimensions per cluster. Thus, the measures CE and RNIA can only be applied on the object grouping. Therefore, we optimized F1 and Accuracy as these are the most meaningful measures where no information is given about the relevant subspaces (cf. Sec. 10.3 and Sec. 10.4.1).

For ease of illustration we picked one real world data set for discussion, while the remaining data sets are given in Figure 10.13. Figure 10.12 shows the results for the *glass* data set. We show minimum and maximum values for various measures, starting with F1 and Accuracy which were used for parameter optimization. For these optimized results, we also show all other measures described in Section 10.3. For each measure we highlighted the best 95% results in gray. Additional information

like the coverage of the resulting clustering, i.e. the proportion of objects which are contained in at least one cluster, the number of clusters, the average dimensionality (cf. Section 10.2), and the runtime are also included in the figures.

Glass (size: 214; dim: 9)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,51	0,31	0,67	0,50	0,02	0,00	0,06	0,00	0,39	0,24	1,00	1,00	6169	175	5,4	3,1	411195	1375
DOC	0,74	0,50	0,63	0,50	0,23	0,13	0,93	0,33	0,72	0,50	0,93	0,91	64	11	9,0	3,3	23172	78
MINECLUS	0,76	0,40	0,52	0,50	0,24	0,19	0,78	0,45	0,72	0,46	1,00	0,87	64	6	7,0	4,3	907	15
SCHISM	0,46	0,39	0,63	0,47	0,11	0,04	0,33	0,20	0,44	0,38	1,00	0,79	158	30	3,9	2,1	313	31
SUBCLU	0,50	0,45	0,65	0,46	0,00	0,00	0,01	0,01	0,42	0,39	1,00	1,00	1648	831	4,9	4,3	14410	4250
FIRES	0,30	0,30	0,49	0,49	0,21	0,21	0,45	0,45	0,40	0,40	0,86	0,86	7	7	2,7	2,7	78	78
INSCY	0,57	0,41	0,65	0,47	0,23	0,09	0,54	0,26	0,67	0,47	0,86	0,79	72	30	5,9	2,7	4703	578
PROCLUS	0,60	0,56	0,60	0,57	0,13	0,05	0,51	0,17	0,76	0,68	0,79	0,57	29	26	8,0	2,0	375	250
P3C	0,28	0,23	0,47	0,39	0,14	0,13	0,30	0,27	0,43	0,38	0,89	0,81	3	2	3,0	3,0	32	31
STATPC	0,75	0,40	0,49	0,36	0,19	0,05	0,67	0,37	0,88	0,36	0,93	0,80	106	27	9,0	9,0	1265	390

Figure 10.12: Real world data set: *glass*

For F1 and Accuracy, the cell-based paradigm shows best results while also two density-based approaches have good accuracy values and one clustering oriented approach has the best result according to the F1 measure. However, going a little more into details, we observe that only DOC, MINECLUS and INSCY have also good values in CE and RNIA. Possible explanations can be derived from the basic measures: CLIQUE and SUBCLU achieve good F1 and Accuracy, but are punished by CE and RNIA for detection of very many clusters (far more than number of objects: 214 in the glass data set). Due to this excessive cluster detection covering 100% of the data (including noise as in most real world data sets) we also observe very high runtimes for both approaches. For our biggest real world data set, *pendigits*, SUBCLU did not even finish. Although SCHISM and STATPC have best results in Accuracy or F1 they show similar low values in CE and RNIA. Here the reason might be the detected relevant dimensions. While SCHISM tends to detect only very low dimensional projections, STATPC detects for all real world data sets full dimensional clusters (except on the *liver* data set).

The main properties we observed on synthetic data are validated by our real world scenarios. The cell-based paradigm overall shows best results with low runtimes in its recent representatives. The distance-based paradigm falls prey to its high runtimes and only finish on small data and only up to medium dimensional data (like e.g. glass with 10 dimensions). Clustering oriented approaches have shown reasonable runtimes and easy parametrization as they directly control the clustering result. However, as they do not define cluster properties explicitly like cell-based approaches, they have lower quality measure values.

10.5 Contributions to the community

With this study we provide a thorough evaluation and comparison of clustering in subspace projections of high dimensional data. We gave an overview of three major paradigms (cell-based, density-based and clustering oriented). We highlighted important properties for each of these paradigms and compared them in extensive

Vowel (size: 990; dim: 10)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,23	0,17	0,64	0,37	0,05	0,00	0,44	0,01	0,10	0,09	1,00	1,00	3062	267	4,9	1,9	523233	1953
DOC	0,49	0,49	0,44	0,44	0,14	0,14	0,85	0,85	0,58	0,58	0,86	0,86	64	64	10,0	10,0	120015	120015
MINECLUS	0,48	0,43	0,37	0,37	0,09	0,04	0,62	0,34	0,60	0,46	0,98	0,87	64	64	7,2	3,6	7734	5204
SCHISM	0,37	0,23	0,62	0,52	0,05	0,01	0,43	0,11	0,29	0,21	1,00	0,93	494	121	4,3	2,8	23031	391
SUBCLU	0,24	0,18	0,58	0,38	0,04	0,01	0,39	0,04	0,30	0,13	1,00	1,00	10881	709	3,6	2,0	26047	2250
FIRES	0,16	0,14	0,13	0,11	0,02	0,02	0,14	0,13	0,16	0,13	0,50	0,45	32	24	2,1	1,9	563	250
INSCY	0,82	0,33	0,61	0,15	0,09	0,07	0,75	0,26	0,94	0,21	0,90	0,81	163	74	9,5	4,3	75706	39390
PROCLUS	0,49	0,49	0,44	0,44	0,11	0,11	0,53	0,53	0,65	0,65	0,67	0,67	64	64	8,0	8,0	766	766
P3C	0,08	0,05	0,17	0,16	0,12	0,08	0,69	0,43	0,13	0,12	0,98	0,95	3	2	7,0	4,7	1610	625
STATPC	0,22	0,22	0,56	0,56	0,06	0,06	0,12	0,12	0,14	0,14	1,00	1,00	39	39	10,0	10,0	18485	16671

Diabetes (size: 768; dim: 8)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,70	0,39	0,72	0,69	0,03	0,01	0,14	0,01	0,23	0,13	1,00	1,00	349	202	4,2	2,4	11953	203
DOC	0,71	0,71	0,72	0,69	0,31	0,26	0,92	0,79	0,31	0,24	1,00	0,93	64	17	8,0	5,1	1E+06	51640
MINECLUS	0,72	0,66	0,71	0,69	0,63	0,13	0,89	0,58	0,29	0,17	0,99	0,96	39	3	6,0	5,2	3578	62
SCHISM	0,70	0,62	0,73	0,68	0,08	0,01	0,36	0,09	0,34	0,20	1,00	0,79	270	21	4,2	3,9	35468	250
SUBCLU	0,74	0,45	0,71	0,68	0,01	0,01	0,01	0,01	0,14	0,11	1,00	1,00	1601	325	4,7	4,0	190122	58718
FIRES	0,52	0,03	0,65	0,64	0,12	0,00	0,27	0,00	0,68	0,00	0,81	0,03	17	1	2,5	1,0	4234	360
INSCY	0,65	0,39	0,70	0,65	0,37	0,11	0,45	0,42	0,44	0,15	0,83	0,73	132	3	6,7	5,7	112093	33531
PROCLUS	0,67	0,61	0,72	0,71	0,34	0,21	0,78	0,69	0,23	0,19	0,92	0,78	9	3	8,0	6,0	360	109
P3C	0,39	0,39	0,66	0,65	0,56	0,11	0,85	0,22	0,09	0,07	0,97	0,88	2	1	7,0	2,0	656	141
STATPC	0,73	0,59	0,70	0,65	0,06	0,00	0,63	0,17	0,72	0,28	0,97	0,75	363	27	8,0	8,0	27749	4657

Pendigits (size: 7494; dim: 16)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,30	0,17	0,96	0,86	0,06	0,01	0,20	0,06	0,41	0,26	1,00	1,00	1890	36	3,1	1,5	67891	219
DOC	0,52	0,52	0,54	0,54	0,18	0,18	0,35	0,35	0,53	0,53	0,91	0,91	15	15	5,5	5,5	178358	178358
MINECLUS	0,87	0,87	0,86	0,86	0,48	0,48	0,89	0,89	0,82	0,82	1,00	1,00	64	64	12,1	12,1	780167	692651
SCHISM	0,45	0,26	0,93	0,71	0,05	0,01	0,30	0,08	0,50	0,45	1,00	0,93	1092	290	10,1	3,4	5E+08	21266
SUBCLU	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
FIRES	0,45	0,45	0,73	0,73	0,09	0,09	0,33	0,33	0,31	0,31	0,94	0,94	27	27	2,5	2,5	169999	169999
INSCY	0,65	0,48	0,78	0,68	0,07	0,07	0,30	0,28	0,77	0,69	0,91	0,82	262	106	5,3	4,6	2E+06	1E+06
PROCLUS	0,78	0,73	0,74	0,73	0,31	0,27	0,64	0,45	0,90	0,71	0,90	0,74	37	17	14,0	8,0	6045	4250
P3C	0,74	0,74	0,72	0,72	0,28	0,28	0,58	0,58	0,76	0,76	0,90	0,90	31	31	9,0	9,0	2E+06	2E+06
STATPC	0,91	0,32	0,92	0,10	0,09	0,00	0,67	0,11	1,00	0,53	0,99	0,84	4109	56	16,0	16,0	5E+07	3E+06

Liver-Dis. (size: 345; dim: 6)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,68	0,65	0,67	0,58	0,08	0,02	0,38	0,03	0,10	0,02	1,00	1,00	1922	19	4,1	1,7	38281	15
DOC	0,67	0,64	0,68	0,58	0,11	0,07	0,51	0,35	0,18	0,11	0,99	0,90	45	13	3,0	1,9	625324	1625
MINECLUS	0,73	0,63	0,65	0,58	0,09	0,09	0,68	0,48	0,33	0,16	0,99	0,92	64	32	4,0	3,7	49563	1954
SCHISM	0,69	0,69	0,68	0,59	0,04	0,03	0,45	0,26	0,10	0,08	0,99	0,99	90	68	2,7	2,1	31	0
SUBCLU	0,68	0,68	0,64	0,58	0,11	0,02	0,68	0,05	0,07	0,02	1,00	1,00	334	64	3,4	1,3	1422	47
FIRES	0,58	0,04	0,58	0,56	0,14	0,00	0,39	0,01	0,37	0,00	0,84	0,03	10	1	3,0	1,0	531	46
INSCY	0,66	0,66	0,62	0,61	0,03	0,03	0,42	0,39	0,21	0,20	0,85	0,81	166	130	2,1	2,1	407	234
PROCLUS	0,53	0,39	0,63	0,63	0,26	0,11	0,66	0,25	0,05	0,05	0,83	0,46	6	2	5,0	3,0	78	31
P3C	0,36	0,35	0,58	0,58	0,55	0,27	0,96	0,47	0,02	0,01	0,98	0,94	2	1	6,0	3,0	172	32
STATPC	0,69	0,57	0,65	0,58	0,23	0,01	0,58	0,37	0,63	0,05	0,77	0,71	159	4	6,0	3,3	1890	781

Shape (size: 160; dim: 17)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,31	0,31	0,76	0,76	0,01	0,01	0,07	0,07	0,66	0,66	1,00	1,00	486	486	3,3	3,3	235	235
DOC	0,90	0,83	0,79	0,54	0,56	0,38	0,90	0,82	0,93	0,86	1,00	1,00	53	29	13,8	12,8	2E+06	86500
MINECLUS	0,94	0,86	0,79	0,60	0,58	0,46	1,00	1,00	0,93	0,82	1,00	1,00	64	32	17,0	17,0	46703	3266
SCHISM	0,51	0,30	0,74	0,49	0,10	0,00	0,26	0,01	0,85	0,55	1,00	0,92	8835	90	6,0	3,9	712964	9031
SUBCLU	0,36	0,29	0,70	0,64	0,00	0,00	0,05	0,04	0,89	0,88	1,00	1,00	3468	3337	4,5	4,1	4063	1891
FIRES	0,36	0,36	0,51	0,44	0,20	0,13	0,25	0,20	0,88	0,82	0,45	0,39	10	5	7,6	5,3	63	47
INSCY	0,84	0,59	0,76	0,48	0,18	0,16	0,37	0,24	0,94	0,87	0,88	0,82	185	48	9,8	9,5	22578	11531
PROCLUS	0,84	0,81	0,72	0,71	0,25	0,18	0,61	0,37	0,93	0,91	0,89	0,79	34	34	13,0	7,0	593	469
P3C	0,51	0,51	0,61	0,61	0,14	0,14	0,17	0,17	0,80	0,80	0,66	0,66	9	9	4,1	4,1	140	140
STATPC	0,43	0,43	0,74	0,74	0,45	0,45	0,55	0,55	0,56	0,56	0,92	0,92	9	9	17,0	17,0	250	171

Breast (size: 198; dim: 33)

	F1		Accuracy		CE		RNIA		Entropy		Coverage		NumClusters		AvgDim		Runtime	
	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min
CLIQUE	0,67	0,67	0,71	0,71	0,02	0,02	0,40	0,40	0,26	0,26	1,00	1,00	107	107	1,7	1,7	453	453
DOC	0,73	0,61	0,81	0,76	0,11	0,04	0,84	0,07	0,46	0,27	1,00	0,80	60	6	27,2	2,8	1E+06	37515
MINECLUS	0,78	0,69	0,78	0,76	0,19	0,18	1,00	1,00	0,56	0,37	1,00	1,00	64	32	33,0	33,0	40359	29437
SCHISM	0,67	0,67	0,75	0,69	0,01	0,01	0,36	0,34	0,35	0,34	1,00	0,99	248	197	2,3	2,2	158749	114609
SUBCLU	0,68	0,51	0,77	0,67	0,02	0,01	0,54	0,04	0,27	0,24	1,00	0,82	357	5	2,0	1,0	5265	16
FIRES	0,49	0,03	0,76	0,76	0,03	0,00	0,05	0,00	1,00	0,01	0,76	0,04	11	1	2,5	1,0	250	31
INSCY	0,74	0,55	0,77	0,76	0,02	0,00	0,24	0,11	0,60	0,39	0,97	0,74	2038	167	11,0	4,4	134373	63484
PROCLUS	0,57	0,52	0,80	0,74	0,51	0,11	0,65	0,43	0,32	0,23	0,89	0,69	9	2	24,0	18,0	703	141
P3C	0,63	0,63	0,77	0,77	0,04	0,04	0,19	0,19	0,36	0,36	0,85	0,85	28	28	6,9	6,9	6281	6281
STATPC	0,41	0,41	0,78	0,78	0,16	0,16	0,33	0,33	0,29	0,29	0,43	0,43	5	5	33,0	33,0	5187	4906

Figure 10.13: Residual real world data sets

evaluations. In a systematic evaluation we used several quality measures and provide results for a broad range of synthetic and real world data.

We provide the first comparison of different paradigm properties in a thorough evaluation. We could show that the evaluated methods for density-based subspace clustering do not scale to very high dimensional data, while the clustering oriented approaches are affected by noisy data resulting in low clustering quality. The recent cell-based approach MINECLUS outperformed, in most cases, the competitors in both efficiency and clustering quality. Surprisingly, the basic approach PROCLUS, in the clustering oriented paradigm, performs very well in our comparison. In contrast, the basic approaches CLIQUE and SUBCLU of the other two paradigms showed major drawback induced by the tremendously large result set. Recent approaches of these paradigms enhanced the quality and efficiency, however, could reach top results only in few cases. Summing up, we show that computing only a small set of relevant clusters like MINECLUS and PROCLUS and pruning most of the redundant subspace clusters achieves best results. Using this key observation in our own research we showed in the previous parts of this thesis how one can tackle major drawbacks of density-based subspace clustering techniques.

In general, our evaluation constitutes an important basis for subspace clustering research as one can compare a broad set of competing techniques but also compare the used evaluation measure. Using this study one can derive several novel challenges not tackled by other subspace clustering approaches. Furthermore, we observe ongoing publications in this area for which our study gives a baseline for future evaluations. Our proposed baseline includes multiple aspects for a fair comparison not only in evaluation studies: First, a common open source framework with baseline implementations for a fair comparison of different algorithms. Second, a broad set of evaluation measures for clustering quality comparison. Third, a baseline of evaluation results for both real world and synthetic data sets with given parameter settings for repeatability. All of this can be downloaded from our website for further research, comparison or repeatability. In the following chapter we describe the un-

derlying open source framework in more details. It is a major contribution for the community as it enables repeatable evaluation for future publications. Furthermore, this common framework ensures applicability of our research in various domains by providing evaluation and exploration of subspace clustering results.

Chapter 11

Evaluation and exploration framework

As the field of clustering in subspace projections is rather young, there has been a lack of comparative studies on the advantages and disadvantages of the different algorithms as described in the previous chapter. Part of the underlying problem is the lack of available open source implementations that can be used by researchers to understand, compare, and extend subspace and projected clustering algorithms. In this chapter, we discuss the requirements for open source evaluation software. We propose OpenSubspace[1], an open source framework that meets these requirements. OpenSubspace integrates state-of-the-art performance measures and visualization techniques to foster research in subspace and projected clustering.

This open source tool provides a basis for repeatability of evaluations due to its common implementation framework. Integrated into the well established WEKA framework it is based on a common database format for benchmark data used in various publications. It provides an easy way of extending our framework by novel subspace clustering algorithms. This gives researches the opportunity for thorough evaluations of their own methods. Furthermore, this framework enables the easy applicability of subspace clustering research to various domains. The included visualization and exploration methods encourage users to browse through their results and extract their desired knowledge. This interactive and visual exploration may open up subspace clustering research to arbitrary application domains.

11.1 Motivation of an open source framework

Comparison studies and repeatability of experimental results are becoming a major issue in the KDD community. SIGKDD conference has initiated repeatability guidelines for publications to enable repeatability of scientific results shown in experiment sections of publications. Therefore, authors are encouraged to provide implementations and data sets. A major issue for such a repeatability and also for comparative studies on the advantages and disadvantages of the different algorithms is the availability of open source implementations. Moreover, it is challenging to compare approaches published without using a common basis, especially for some young research areas

[1] http://dme.rwth-aachen.de/OpenSubspace/

where evaluation measures and open source tools to evaluate and explore the mining results are not yet available.

In this work we propose such an open source tool for clustering in subspaces of high dimensional data. It tackles main challenges in evaluation and exploration of clustering in subspace projections as a young research area. We focus on two different ways of analyzing subspace clustering results. In recent publications, labeled data is usually used to evaluate the performance. While this provides the possibility of measuring the performance of clustering algorithms, the base assumption that clusters reflect the class structure is not necessarily valid. On the other hand, some approaches resort to the help of domain experts in judging the quality of the result [KKK04, BKKK04, KKRW05]. Where domain experts are available, which is clearly not always the case, they provide very realistic insights into the usefulness of a clustering result. Still, even if this insight is necessarily subjective and not reproducible by other researchers, for subspace clustering there are no tools available for exploring such result set by visualization methods.

Furthermore, as there is no ground truth, nor accepted benchmark data or measures for evaluating subspace and projected clustering, the experimental evaluation can be hardly set into relation to other published results. Especially the results are incomparable, as there are no publicly available common implementations neither for subspace/projected clustering algorithms nor for evaluation measures (cf. Fig 11.1). As a consequence, progress in this research area is slow, and general understanding of the advantages and disadvantages of different algorithms is not established. The source code for experimental evaluation is most of the time implemented by the authors themselves and often not made available to the general public. This hinders further experimental study of recent advances in clustering. As tedious re-implementation is often avoided, only few comparisons between new proposals and existing techniques are published.

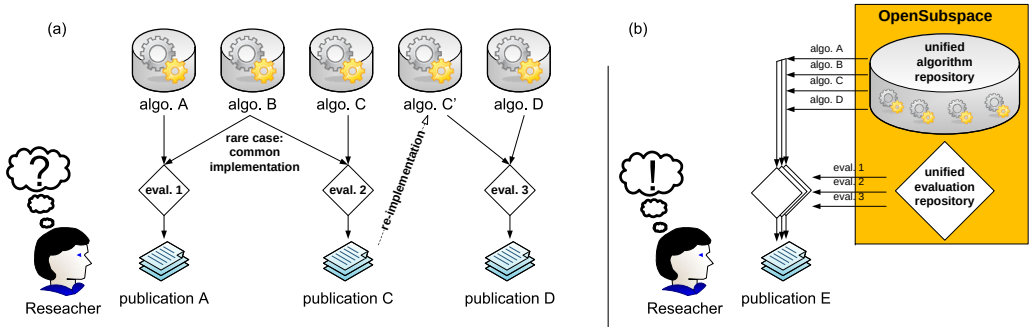


Figure 11.1: Subspace and projected clustering research with and without a general repository for data, algorithms, comparison, and evaluation.

For clustering (and also classification and association rule mining), the open source tool WEKA (Waikato Environment for Knowledge Analysis) has been very helpful in allowing researchers to analyze the behavior of different algorithms in comparison [WF05]. It provides measures for comparison, visualization of the results, and lets researchers add their own algorithms and browse through the implementation of

other techniques.

For subspace and projected clustering, such a general tool does not exist. In this chapter, we discuss the requirements for a successful open source tool that supports evaluation and exploration of subspace and projected clustering algorithms and their cluster results. Our framework OpenSubspace fulfills these requirements by integration of measurement and visualization techniques for in-depth analysis. Furthermore, it will be useful in establishing benchmark results that foster research in the area through better understanding of advantages and disadvantages of different algorithms on different types of data. It includes successful techniques in demonstration systems for visualization and evaluation of subspace and projected clustering [MAK+08, AMK+08].

As anyone will be able to see the implementation, the code base can be continually revised and improved. Researchers may analyze the algorithms on a far greater range of parameter values than would be possible within the scope of a single conference or journal publication (cf. Fig 11.1).

For scientific publications the open source implementations in OpenSubspace enables more fine grained discussions about competing algorithms on a common basis. For authors of novel methods OpenSubspace gives the opportunity to provide their source code and thus deeper insight into their work. This enhances the overall quality of publications as comparison is not based any more on incomparable evaluations of results provided in different publications but on a common algorithm repository with approved algorithm implementations. In Figure 11.1 we compare the current situation (on the left side) with the improved situation having a common repository of both subspace/projected clustering and evaluation measures (on the right side). Thus, OpenSubspace aims at defining a common basis for research and education purposes maintained and extended by the subspace/projected clustering community.

None of the existing data mining frameworks provide both subspace/projected clustering as well as the full analytical and comparative measures for the full knowledge discovery cycle. KNIME (Konstanz Information Miner) is a data mining tool that supports data flow construction for knowledge discovery [BCD+09]. It allows visual analysis and integration of WEKA. Orange is a scripting or GUI object based component system for data mining [DZL04]. It provides data modeling and (statistical) analysis tools for different data mining techniques. Rattle (the R Analytical Tool To Learn Easily) is a data mining toolkit that supports statistical data mining based on the open source statistical language R [Wil08]. Evaluation via a number of measures is supported. In all of these frameworks subspace clustering or projected clustering are not included. ELKI (Environment for DeveLoping KDD-Applications Supported by Index Structures) is a general framework for data mining [AKZ08]. While it also includes subspace and projected clustering implementations, the focus is on index support and data management tasks. With respect to evaluation and exploration, it lacks evaluation measures and visualization techniques for an easy comparison of clustering results. Furthermore, as a stand alone toolkit it does not provide an integration into popular tools like WEKA.

11.1.1 Clustering in subspace projections

While *subspace clustering* and *projected clustering* are rather young areas that have been researched for only one decade, several distinct paradigms can be observed in the literature. Our open source framework includes representatives of these paradigms to provide an overview over the techniques available. We provide implementations of the most recent approaches from different paradigms (cf. Fig. 11.2):

Subspace clustering Subspace clustering was introduced in the *CLIQUE* approach which exploits monotonicity on the density of grid cells for pruning [AGGR98]. *SCHISM* [SZ04] extends *CLIQUE* using a variable threshold adapted to the dimensionality of the subspace as well as efficient heuristics for pruning. Both are grid-based approaches which discretize the data space for efficient detection of dense grid cells in a bottom-up fashion.

In contrast, density-based subspace clustering defines clusters as dense areas separated by sparsely populated areas. In *SUBCLU*, a density monotonicity property is used to prune subspaces in a bottom-up fashion [KKK04]. *PreDeCon* extends this paradigm by introducing the concept of subspace preference weights to determine axis parallel projections [BKKK04]. A further extension *FIRES* proposes an approximative solution for efficient density-based subspace clustering [KKRW05]. In *DUSC*, dimensionality bias is removed by normalizing the density with respect to the dimensionality of the subspace as described in Chapter 2. Its extension *INSCY* focuses on efficient in-process removal of redundancy (cf. Chapter 6). Further developments out of this thesis will be included into the publicly available version as soon as possible.

Projected clustering Projected clustering approaches are partitioning methods that identify disjoint clusters in subspace projections. *PROCLUS* extends the k-medoid algorithm by iteratively refining a full-space k-medoid clustering in a top-down manner [AWY+99]. *P3C* combines one-dimensional cluster cores to higher dimensional clusters bottom-up [MSE06]. Its extension *StatPC* searches for non-redundant significant regions [MS08]. *DOC* a randomized approach using a Monte Carlo algorithm to find projected clusters represented by dense hypercubes [PJAM02] and *MineClus* an extension using the FP-tree for iterative projected clustering [YM03].

Overall, our repository of algorithms forms a basis for comparable experiments on a broad set of different methods.

11.1.2 Challenges in evaluation and exploration

Both subspace clustering and projected clustering pose new challenges to the mining task but especially to evaluation and exploration of the actual clustering results. In the following, we will show that these challenges have not yet been addressed by recent open source systems. Furthermore, they can not be solved by simply applying traditional techniques available for low dimensional clustering paradigms.

The WEKA framework provides several panels for different steps in the knowledge discovery cycle as well as for different data mining tasks (cf. Fig. 11.3). Besides

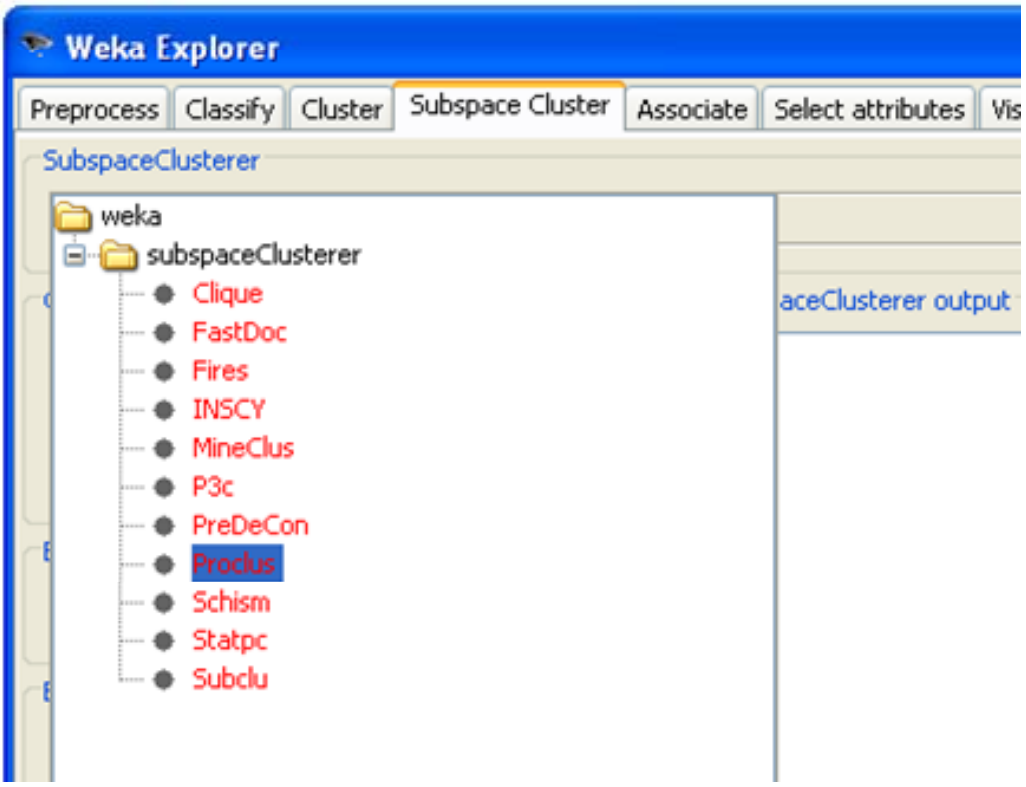


Figure 11.2: Implemented algorithms in OpenSubspace

structuring the GUI for users of the framework, the API reflects these different tasks in being structured according to classifiers, clustering algorithms, etc. This means that the Java class hierarchy reflects the common properties of each of the tasks. From this, several challenges arise in introducing a new data mining task, namely subspace/projected clustering, and new evaluation and visualization methods.

Due to special requirements in high dimensional mining we cannot simply extend the clustering panel in WEKA by adding new algorithms. We have to set up a new subspace panel by introducing techniques specialized to the new requirements in all areas (mining, evaluation and visualization). Subspace and projected clustering algorithms differ from clustering (or other data mining tasks such as classification) in that each cluster is associated with a possibly different subspace projection. As a consequence, the existing representation that assumes that all objects are clustered with respect to the initially chosen dimensions, is not valid. Moreover, the result is not necessarily partitioning. A single object may be part of several subspace clusters. These two aspects are important for the subspace/projected clustering panel, i.e. for the interface that describes common properties of these approaches. Moreover, these aspects have to be taken into consideration for evaluation and visualization as well. Following the same rationale, it is necessary to provide new APIs to allow meaningful analytical and visual tools in the OpenSubspace framework. Any measure that supports subspace and projected clustering evaluation needs to incorporate information on the respective subspace projection of each cluster. Visualization techniques to provide techniques for comparison of results in differing projections, as in

[AKMS07b], cannot plug into existing visualization interfaces for traditional clustering in WEKA for the same reason.

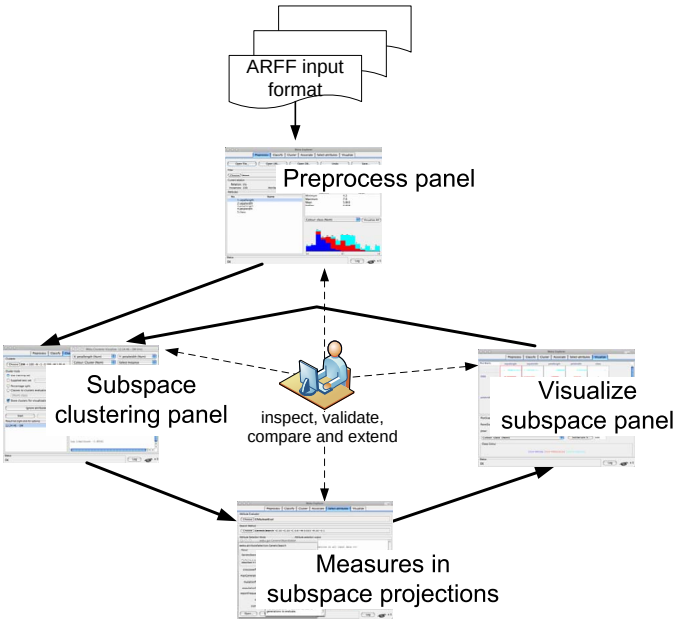


Figure 11.3: KDD cycle in WEKA for subspace and projected clustering

As a consequence, OpenSubspace allows identifying any result with a corresponding set of dimensions, i.e. the subspace in which the result cluster resides. This is taken into consideration both for the subspace/projected clustering panel itself with the display of numeric results and evaluation measures, as well as for the visualization panel. This streamlined approach ensures that for all steps in the KDD cycle, representation in the correct subspace projection is achieved.

11.2 OpenSubspace framework

With OpenSubspace we provide an open source framework tackling the challenges mentioned in the previous section. By fully integrating OpenSubspace into the WEKA framework we build on an established data mining framework covering the whole KDD cycle: pre-processing, mining, evaluation and visualization of the results, additionally including user feedback to the mining algorithm to close the KDD cycle. With OpenSubspace we focus on the mining, evaluation and exploration steps in this cycle (cf. Fig. 11.4). Providing a common basis for subspace/projected clustering as a novel mining step we achieve a framework for fair comparison of different approaches. For evaluation and exploration of the subspace and projected clusters OpenSubspace provides various evaluation measures for objective comparison of different clustering results. Furthermore, OpenSubspace provides visualization methods for an interactive exploration. Please refer to our website where we document our ongoing work in this project. It also contains more detailed information about OpenSubspace, its usage

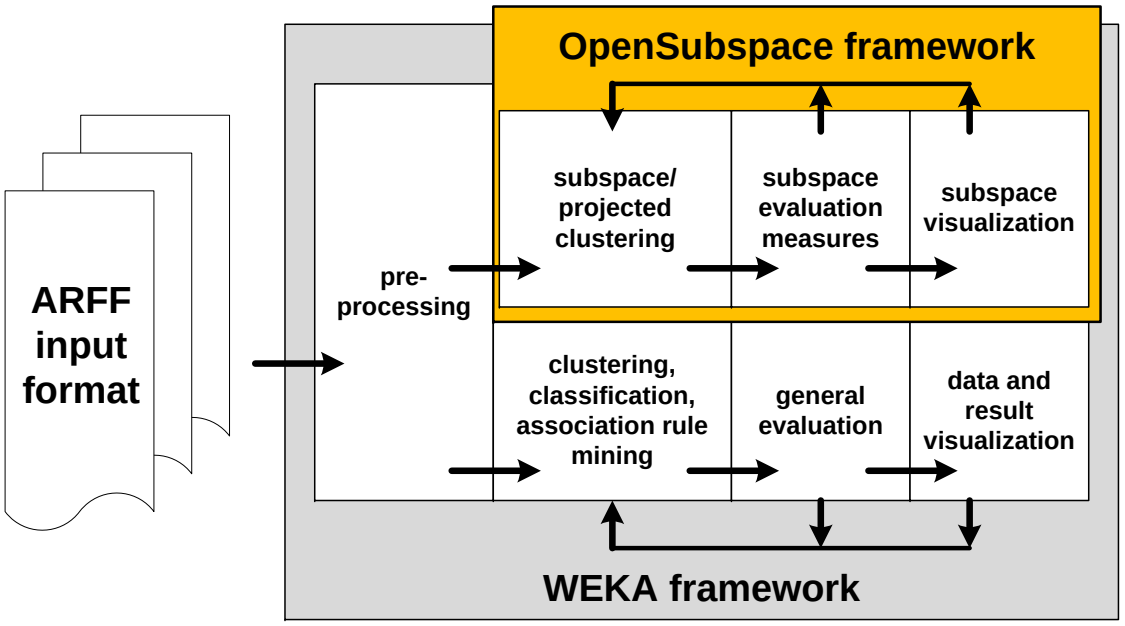


Figure 11.4: KDD cycle of OpenSubspace in WEKA

and extension. In the following, we will give an overview on the major contributions of OpenSubspace to the subspace clustering community:

- Transparency of implementations
- Evaluation and comparison of algorithms
- Extensibility to further approaches

As we will show open source is the key property for all of these contributions. Open source code enables us to compare and validate the correctness of algorithm implementations. It gives us the basis for evaluating existing approaches on a common basis and leads thus to a fair comparison in future publications. By having the code of recent approaches at hand we enable the extension of existing algorithms to everyone and not only to the authors of these approaches.

11.2.1 Transparency of implementations

The basis for thorough and fair evaluation is a common basis for all implementations. OpenSubspace provides such a basis for data access supporting both main and secondary storage. Furthermore, the framework provides a common interface for subspace clustering implementations. Algorithms which extend this interface can be easily plugged into OpenSubspace.

All of these algorithms are provided as open source. This transparency of the underlying implementation ensures high quality algorithms in the framework. The research community is able to review these implementations according to the original publication of the algorithm. Even improved versions can be provided which go beyond the descriptions in the publications using novel data structures, heuristics or

approximation for specialized purposes. The benefit of reviewing implementations based on open source is especially useful as in most publications algorithms can only sketched by the authors. This makes it difficult to re-implement such approaches. Various different interpretations of one approach could arise if only closed implementations were available. Using these different implementations of the same approach leads to incomparable results in scientific publications as evaluations have different bases. Open source repositories as in OpenSubspace prevent such differing implementations. Overall OpenSubspace aims at a transparent and thus fair basis for evaluations of various approaches for detecting meaningful subspace clusters.

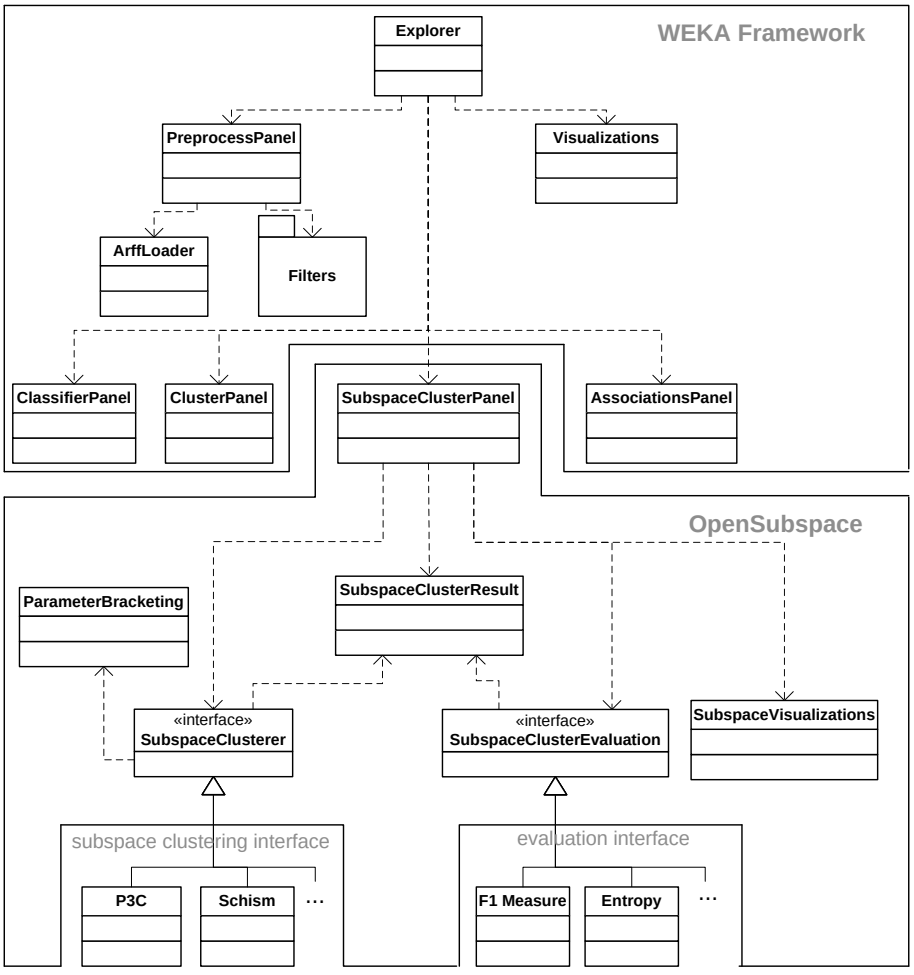


Figure 11.5: UML class diagram of the OpenSubspace framework

For extending the OpenSubspace algorithm repository our framework incorporates two open interfaces, which enable extensibility to further subspace clustering algorithms and new evaluation measurements. In Figure 11.5 we show the main classes of our OpenSubspace framework which extends the WEKA framework by a new subspace clustering panel.

Subspace clustering shows major differences compared to traditional clustering; e.g. an object can be part of several subspace clusters in different projections. We therefore do not extend the clustering panel, but provide a separate subspace clus-

tering panel.

Recent subspace clustering algorithms described in Section 11.1.1 are implemented based on this framework. The abstraction of subspace clustering properties in OpenSubspace allows to easily add new algorithms through our new subspace clustering interface.

11.2.2 Evaluation and comparison of algorithms

Given the framework with transparent implementations of subspace clustering algorithms OpenSubspace enables researchers to evaluate their methods against competing approaches available in our repository. We provide a basis for developing new methods to perform an objective evaluation on arbitrary subspace clustering algorithms. OpenSubspace defines evaluation measurements based on labeled data sets. Measurements like entropy, coverage, F1-value, Cluster Error, RNIA and accuracy used in recent subspace/projected clustering publications form the basis for a thorough evaluation (cf. Fig. 11.6).

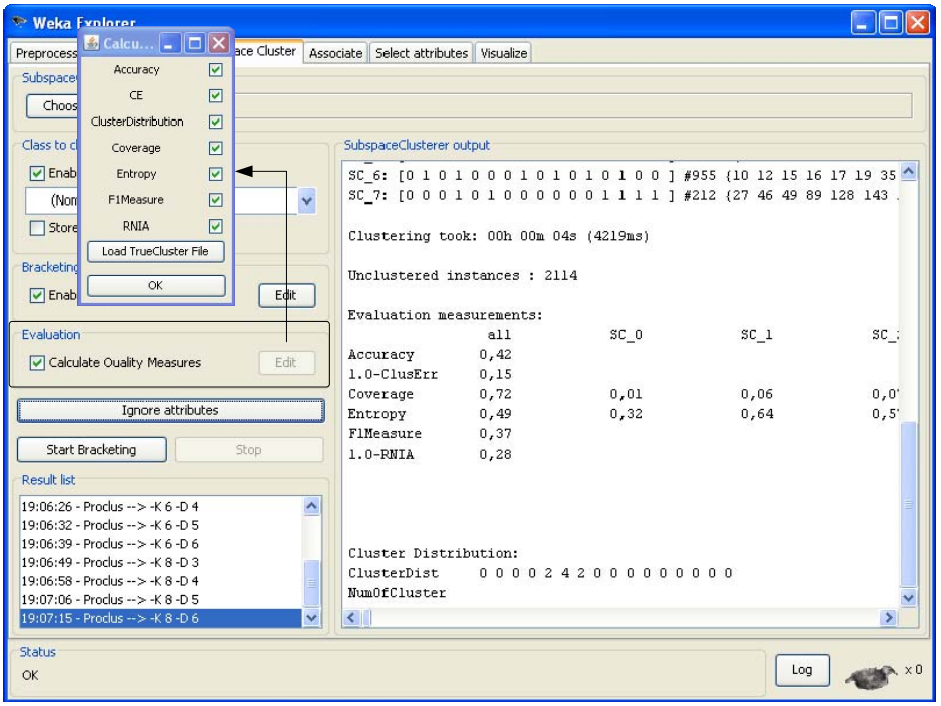


Figure 11.6: Evaluation in OpenSubspace

All evaluation methods are provided as open source as well. For a fair and comparative evaluation these measurements have to be accessible to all researchers. Review and refinement of these measurements is essential as there is always the possibility of different interpretations of these measures. As a ground truth is not given for subspace clustering the data mining community has to develop new evaluation measures that rate the quality of different approaches. This seems to be as difficult as the mining task itself. Therefore, we do not only provide several evaluation techniques to measure the quality of the subspace clustering, but also an open interface

(cf. Fig. 11.5) to implement new measures. Further measures can be added by our new evaluation interface, which allows to define new quality criteria for subspace clustering on a common basis for all algorithms.

Overall, OpenSubspace provides the framework for using several, widely used, evaluation measures for subspace and projected clustering algorithms. In Figure 11.6 we present the evaluation output with various measures for comparing subspace clustering results. This allows easy extension of published results for various measures and direct comparison. Over time, as more and more results are available on different datasets and with respect to different evaluation measures, a benchmark background is built. It provides the means for in-depth understanding of algorithms and evaluation measures and fosters research in this area based on individual researcher's findings.

11.2.3 Exploration of subspace clustering results

Evaluation measures summarize the result set in typically one real valued rating; however, visualization of results for more insight might be interesting. OpenSubspace, therefore, includes specialized visualizations for subspace clustering results with the possibility for interactive exploration [MAK+08]. As stated before, subspace and projected clustering algorithms typically provide overwhelming result sets. Investigating these results is sometimes as difficult as looking at the raw data. For some specialized or domain dependent mining tasks it is even more important to investigate the actual clustering than to compare it with competing approaches. OpenSubspace provides specialized visualization techniques which close the KDD cycle by providing user feedback (cf. Fig. 11.4) [MAK+08]. Our framework provides interactive exploration of the results and thus the opportunity to refine the mining step by exploring different parameter settings and their resulting clustering output.

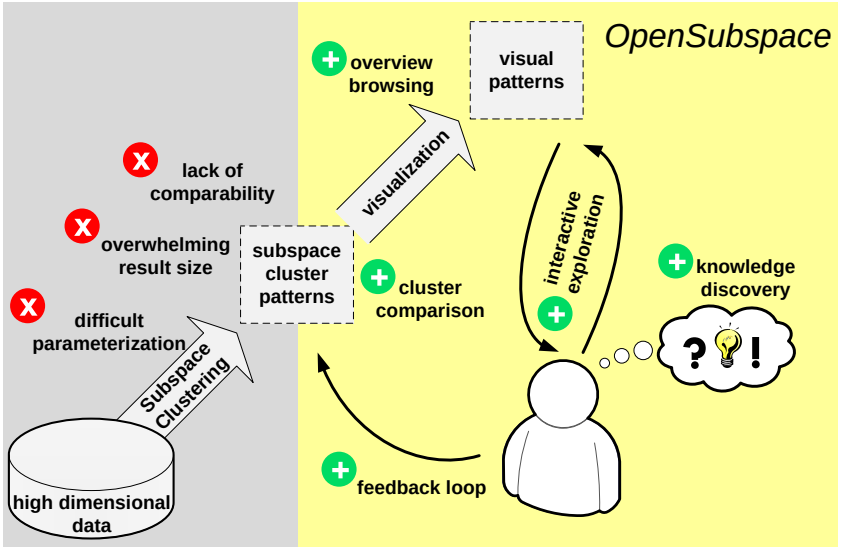


Figure 11.7: Closing the KDD cycle in OpenSubspace

In our tool we use the cognitive abilities of human users to transfer the detected subspace cluster patterns to actual knowledge about the high dimensional data an-

alyzed. Thus, OpenSubspace completes the final steps of the KDD (Knowledge Discovery in Databases) process for subspace clustering (cf. Figure 11.7). In the following, we describe how OpenSubspace supports the knowledge discovery process through visualization and interactive exploration of subspace clusterings. Users may browse an overview of the entire subspace clustering, analyze subspace cluster characteristics in-depth and zoom into object groupings. Bracketing of different parameter settings enables users to immediately see the effects of parameters and to provide feedback to further improve the subspace clustering. Furthermore, OpenSubspace may serve as a teaching and exploration tool for the data mining community to visually assess different subspace clustering paradigms.

11.2.4 Visualization techniques

OpenSubspace provides visualization techniques to present subspace clustering results such that users can easily gain an overview of the detected patterns, as well as an in-depth understanding of individual subspace clusters and their mutual relationship.

Gaining a meaningful overview is crucial in allowing users to assess the overall subspace clustering result. As mentioned before, subspace clustering is inherently challenging as both the typical number of resulting subspace clusters is usually enormous as well as that clusters in different projections are difficult to understand. Visualization techniques that were developed for full space clustering results rely on a common representation, i.e. no subspace projections [AKMS07b]. Consequently, they cannot be applied to subspace clustering.

Our framework thus contains specialized techniques for visualization of subspace clustering. 2d and 3d models are an adequate representation for human cognitive abilities. Based on a recently developed comparison measure for subspace clusters our system provides an overview on the entire subspace clustering result by MDS (multidimensional scaling) plots in both 2d and 3d [AKMS07b]. As illustrated in Figures 11.8, MDS approximates distances in high dimensional spaces by two or three dimensions. While the 2d representation is a static view that allows for easy reading, the 3d MDS plots allow users to interactively explore the overall subspace clustering result. They may move around the 3d representation to focus on those subspace clusters they are most interested in. At any point, they may choose individual subspace clusters in the plot to obtain more detailed information.

Thus, our MDS plots provide an overview on subspace clustering. Moreover, it helps users in interactively determining the best parameter setting. For any subspace clustering algorithm, some core parameters tend to have a large influence on the resulting output. We therefore present a bracketing representation, i.e. a series of 2d MDS plots for different parameters. Users thus get a clear visual impression of the effect of parameters and may choose the best ones for a feedback loop that generates the desired subspace clustering.

For in-depth analysis of any subspace clustering algorithm, representation of the key features of subspace clusters in a cognitively meaningful way is crucial. As subspace clustering results represent patterns in different projections by their very nature, visualization should contain information on the respective subspaces, the

cluster values and additional information on the interestingness measures computed by the subspace clustering algorithm. We use a color-coding scheme where the different axis in the HSV color space are used to represent different aspects of subspace clusters in a very compact and easy to understand manner [AKMS07b]. For easy navigation, subspace clusters can be zoomed into, and understood using a color legend on values of the subspace clusters.

11.2.5 Interactive exploration

The conceptual design for interactive exploration in OpenSubspace is based on the *Visual Exploration Paradigm* [Kei02]: Starting from an overview over the subspace clustering result the user can navigate through the visualized patterns. By selecting subjectively interesting subspace clusters, the user may then obtain more detailed information where desired. Detailed information is provided on three levels: for entire subspace clusterings, for single subspace clusters, as well as for individual objects. Based on the discovered knowledge, the user can give feedback to the system for further improved results. This feedback loop enables the system to use the cognitive abilities of humans for better parameter settings and thus for meaningful subspace clusters.

Overview Browsing. Interactive exploration starts from an overview of all mined subspace clusters in which the user can browse. The automatically detected patterns are thus presented to the user for a general impression of the result and a comparison of the resulting subspace clusters. As clusters are detected in arbitrary subspaces, they cannot be compared based on the full space dimensionality. We thus incorporate a distance function that takes the main characteristics of subspace clusters, their subspace dissimilarity and object dissimilarity into account for visualization in an MDS plot [AKMS07b]. Based on such an overall distance function, subspace clusters can be intuitively represented as circles in a 2d or 3d space (Figure 11.8). This approximation of the original high dimensional information to a 2d or 3d representation allows human users to easily understand the result just by the visual impression. We enrich this MDS information by additional visual features. The size of a subspace cluster is represented as the diameter of the circle. Its dimensionality is encoded by the color of the circle. This information allows users to identify similar subspace clusters, those clusters of similar dimensionality, or of similar size, or to study the overall distribution of these characteristics in the result for further analysis.

Parameter Bracketing. Parameter setting is in general a difficult task for most data mining algorithms. This is especially the case for unsupervised techniques like clustering, where typically no prior knowledge about the data is available. This inherent problem of clustering is even more present in subspace clustering as the user has to provide parametrization for detecting clusters in different subspaces. In general,

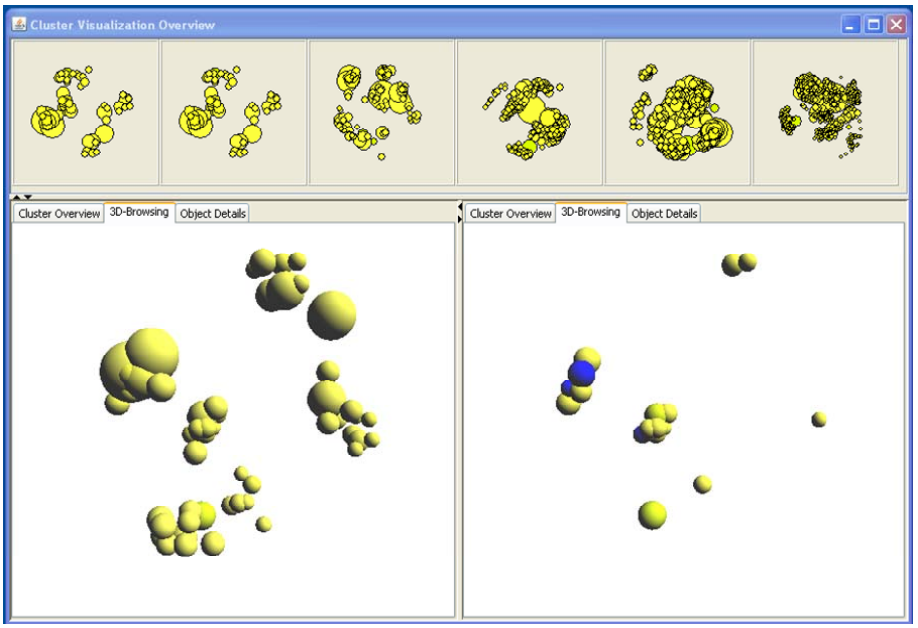


Figure 11.8: Visualization in OpenSubspace

the problem can be solved by guessing a parameter setting, looking at the result and then trying to choose a better parameter setting. To speed up this tedious process for users and give them more information to base their parameter choice on, we visualize a set of different subspace clusters at once, called bracketing of different parameter settings for direct feedback. This means that users obtain a series of MDS plots (cf. upper part of Figure 11.8) from which they pick the most appropriate one(s) for subsequent runs of the subspace clustering algorithms. By directly comparing the results of different parameter settings, parametrization is no longer a guess, but becomes an informed decision based on the visual analysis of the effects of parameters. Moreover, this process is far more comfortable for users and allows reaching the desired subspace clustering result in fewer steps.

Direct subspace cluster comparison. For a more detailed analysis of two different parameter settings the user can select two clusterings out of the presented series of MDS plots by clicking on them in the bracketing representation. These two subspace clusterings are then presented as larger plots in the lower part of the cluster overview screen. Once again, detailed information for the subspace clusters can be obtain by picking individual subspace clusters.

3d Browsing. For the overview browsing we provide static 2-dimensional MDS plots. These static views provide a fixed perspective for easy comparison. For in-depth browsing, where focusing to different parts of the subspace clustering is of interest, a flexible navigation through MDS plots is provided. 3-dimensional MDS

plot browsing allows users to shift, rotate and zoom into the MDS plot using standard 2-dimensional input devices or 3-dimensional input devices that allow for even more intuitive navigation. The user may thus identify similar or dissimilar subspace clusters that are of specific interest. In Figure 11.8, we show two 3-dimensional MDS plots representing two clustering results.

11.2.6 Extensibility of OpenSubspace

As a novel framework OpenSubspace provides the basis for further research. There are several algorithms implemented in our subspace/projected clustering repository. For evaluation measures we have included recently used measures in this field [SZ04, MSE06, BZ07]. However, as subspace clustering has just started to become a broader research topic, these evaluation measures can be only seen as first steps that are likely to be extended greatly in the near future. We included different visualization techniques in OpenSubspace which we presented in a recent demonstration system [MAK+08].

All three areas (mining, evaluation and visualization with interactive exploration) can be extended by open interfaces. Due to the fact that the whole framework is given as open source code it is easy to develop new algorithms, evaluation measures and visualizations. For researchers who wish to develop their own novel algorithm in this field we provide an easy way to integrate their approach into our framework and to perform a fair evaluation with competing approaches. Thus it is a key property of OpenSubspace to define an open basis for the development of new approaches, evaluation and visualization techniques.

We used and still use our framework for subspace clustering research but also for education in advanced data mining courses. In both cases we got positive feedback from our students who enjoyed easy and wide access and the predefined interfaces in our framework. Furthermore, we got encouraging feedback also by the community for our recent demonstration system which integrates extensible mining techniques into WEKA [AMK+08].

11.3 Enabling repeatability in future research

With OpenSubspace we provide an open source framework for the emerging research area of clustering in subspace projections. The aim of our framework is to establish a basis for comparable and repeatable experiments and thorough evaluations in the area of clustering on high dimensional data. OpenSubspace is designed as the basis for comparative studies on the advantages and disadvantages of different subspace/projected clustering algorithms. Providing OpenSubspace as open source, our framework can be used by researchers and educators to understand, compare, and extend subspace and projected clustering algorithms. The integrated state-of-the-art performance measures and visualization techniques are first steps for a thorough evaluation of algorithms in this field of data mining.

As major benefit OpenSubspace ensures repeatability of results in scientific publications. Based on a common framework all implementations are available and can be used by any researcher to compare against previous techniques. As integrated in the well established WEKA framework our OpenSubspace tool is already used by several international scientists for their research work. Especially for our research it is widely used in evaluations. All experiments shown in this thesis are based on this framework such that we can ensure repeatability of our results. Furthermore, we use this framework in lab courses to provide students an easy way of rapid prototyping as well as in lectures to illustrate the effects of subspace clustering algorithms on toy databases.

Overall, the OpenSubspace framework can be seen as the natural basis for our future research in this area. We are currently developing evaluation measures that meet the requirements for a global quality rating of subspace clustering results. Evaluations with the given measurements show that none of the measurements can provide an overall rating of quality. Some measurements give contradicting quality ratings on some data sets. Such effects show us that further research should be done in this area.

Visualization techniques give an overall impression on the groupings detected by the algorithms. However, further research of meaningful and intuitive visualization is clearly necessary for subspace clustering. The open source implementations of subspace/projected mining algorithms and the framework might encourage also researches in Visual Analytics or Human Computer Interaction to develop more meaningful visualization and exploration techniques.

For an overall evaluation framework OpenSubspace provides algorithm and evaluation implementations. However, further work has to be done to collect a bigger test set of high dimensional data sets. On such a benchmarking set one could collect best parameter settings for various algorithms, best quality results and screenshots of subspace clustering result visualizations as example clusters on these data sets. The aim of an overall evaluation framework with benchmarking data will then lead to a more mature subspace/projected clustering research field in which one can easily judge the quality of novel algorithms by comparing it with approved results of competing approaches.

Part IV

Outlier mining in subspace projections

Chapter 12

Outlier ranking based on subspace clustering results

In the previous parts of this thesis the focus was on detection of clusters as groups of similar objects. In contrast to these groups, outliers are single objects highly deviating from the other objects. For some applications, such outliers should not be considered as noise (like in clustering) but as interesting patterns. Thus, outlier mining is an important data analysis task which distinguishes exceptional outliers from regular objects. It has shown to be essential for fraud detection, consistency checks, anomaly detection and many more where users are interested in these exceptional objects sorted by their degree of deviation. There are well established methods which are successful in measuring the degree of deviation for outlier ranking. However, similar to traditional clustering approaches, these outlier methods fail in high dimensional spaces as they measure deviation in the full space. Outliers are hidden in subspace projections and do not show up in the scattered full space. Thus, in the following we focus on ranking highly deviating objects hidden in subspaces. Based on our subspace clustering research we propose three novel outlier ranking methods.

First, we use detected subspace clusters as input for outlier ranking. Given a subspace clustering result we propose a post-processing to extract outliers hidden in subspaces. It is only a visionary work in the area of outlier ranking, but we derived several challenges for the following chapters. As second approach, we propose a direct method for outlier ranking in subspace projections without prior clustering in Chapter 13. We tackle several challenges by our adaptive outlier ranking method showing high quality outlier detection compared to competing approaches. And third, in addition to this ranking of objects we propose novel descriptive components providing knowledge about the underlying outlier properties in Chapter 14.

In this chapter, we first focus on outlier ranking as post-processing to subspace clustering. We propose our OutRank approach for ranking outliers in heterogeneous high dimensional data. As we have described in Chapter 7 heterogeneous (continuous and categorical) data poses additional challenges to mining high dimensional databases. As these challenges apply also for outlier ranking, we base on our previous work having tackled these challenges for subspace clustering. We develop novel scoring functions which transform the analyzed structure of the data to a meaning-

ful outlier ranking. In our experiments, we show promising results that indicate the potential of our method for successful outlier ranking in high dimensional data.

12.1 Motivation of outlier ranking based on subspace clustering

Outlier detection is an important data mining task for detection of exceptional objects in today's huge amounts of application data. Applications include consistency checks of sensor network measurements, fraud detection in financial transactions, emergency detection in health surveillance and many more. In general, outliers are objects that deviate from the rest of the data to a great extent. Traditional distance-based [KNT00] or cluster-based [EKSX96, HXD03] outlier mining algorithms suffer from difficulties in parametrization, as the extent of deviation is usually hard to quantify. This has led to outlier ranking based on their degree of deviation, e.g. as in the local outlier factor (LOF) approach [BKNS00] or in its extension to a top-n outlier detection [JTH01]. While these approaches have been successful in low dimensional data, high dimensional and heterogeneous data still pose a challenge to outlier detection. For high dimensional data as prevalent in many application databases, distances grow more and more alike due to the "curse of dimensionality" [BGRS99]. Thus, traditional outlier detection methods fail to separate outliers from the remaining data. Similar to clusters also for outliers hidden in subspace projections, global dimensionality reduction techniques like principal components analysis (PCA) [Jol86] are not adequate. In clustering, the effect of locally varying relevance of attributes has led to the development of subspace clustering techniques. As subspace clustering detects locally relevant attributes for each cluster very effectively, we base our outlier mining approach on subspace clustering results.

For outlier ranking on high dimensional databases, we propose exploiting subspace clustering analysis. A major challenge lies in the fact that usually even outliers will be part of at least one of the exponentially many subspace clusters. Thus, the deviation has to be measured with respect to the prevailing subspace cluster patterns in the data. An additional challenge arises from heterogeneity, i.e. both continuous and categorical attributes, where existing outlier mining approaches usually focus on just one type of attributes.

In this work we propose *OutRank*, an approach that is capable of handling heterogeneous high dimensional data. Overall, we observe similar challenges due to high dimensional and heterogeneous data as already tackled in our subspace clustering model described in Chapter 7. Thus, in this outlier mining approach we propose using subspace clusters for outlier detection. Using any given subspace clustering result we derive an outlier ranking. Thus, any improvement in subspace clustering should yield also enhanced outlier detection. The difficulty lies in meaningful ranking of outliers with respect to possibly overlapping clusters in arbitrary subspaces. As any object may belong to several clusters in one or more subspaces, we define novel scoring functions based on these subspace clusters in the following section. We there-

fore base on a recent subspace clustering model for heterogeneous data (cf. Chapter 7). Preliminary experiments for heterogeneous data in this chapter show that our algorithm outperforms LOADED [GOP04] a link-based approach for heterogeneous data. A thorough evaluation compared to other competing approaches applicable only on continuous data is presented in the following chapters. Furthermore, we demonstrate future research potential toward a general framework for outlier ranking based on subspace clustering results.

12.2 Scoring functions for outlier ranking

Using subspace clustering to analyze the structure of the data allows our approach to deal with high dimensional data. However, compared with traditional full space clustering algorithms like DBSCAN [EKSX96] there is no direct outlier output. Subspace clustering does not compute a partitioning of the data in several clusters and a group of outlier objects. Instead the result is a set of overlapping clusters which typically encloses all objects. In contrast, projected clustering which provides such a partitioning for high dimensional data is not appropriate for outlier ranking. By using projected clustering for outlier detection one might miss outliers deviating in one subspace while being clustered in some other subspace. We thus focus on defining scoring functions as a transformation based on subspace clusters. Scoring must reflect the deviation of objects such that a ranking of outliers can be computed by sorting objects in ascending order of their scores.

In subspace clustering, objects might be clustered in multiple subspaces. Thus, objects are typically in at least one subspace cluster, because in low dimensional subspaces (e.g. one or two attributes) it is most probable to find similar objects. Additionally, objects often belong to more than one subspace cluster; because of the large number of different subspaces, it is likely that each object is similar to other objects in at least one subspace. Thus, we define outliers as objects that are found in abnormally few or low dimensional subspace clusters. For a toy example with three hidden outliers we have depicted three projections of the data and the resulting scoring in Figure 12.1. The detected subspace clusters can be used to indicate the hidden outliers as they are only part of clusters in low dimensional projections. Please note that we use our non-redundant subspace clustering excluding redundant low dimensional clusters as proposed in Chapter 6. Thus, one of the hidden outliers is not part of any cluster. Given the three highlighted subspace clusters we can derive an outlier ranking as depicted in Figure 12.1.

We base on the general observation that outliers are found only in small or low dimensional subspace clusters. Consequently, we develop novel scoring functions based on the result set of subspace clustering. For each object in the database these scoring functions assign positive score values for each object in each subspace cluster. The lower the score, the greater the deviation. Scoring functions should weight these clusters by their size with respect to both number of objects and number of attributes. This reflects the idea that larger subspace clusters in more attributes are stronger witnesses for an object's "normality".

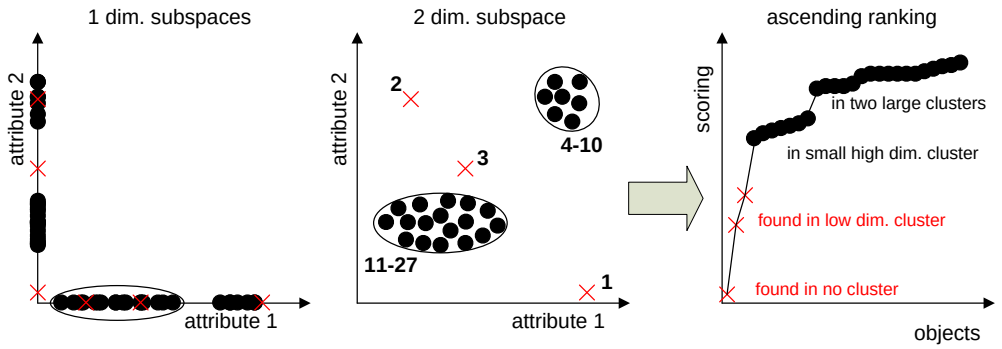


Figure 12.1: Subspace clusters indicating outliers hidden in subspaces

As depicted in Figure 12.1 the first object in our outlier ranking is not part of any cluster. Sorting the scores in ascending order as depicted on the right side of Figure 12.1, it is detected as the most promising outlier. The second and third object are part of one cluster, but show significantly lower score than the residual objects. As both objects are clustered only in one low dimensional cluster all residual objects have higher score due to their stronger witnesses in the two dimensional subspace clusters. Further enhancement of this outlier ranking can be achieved by considering properties like the density of each object. Using density of objects, the third hidden outlier achieves a slightly higher score than the second outlier. This property is included in our second scoring function.

In our first scoring function we incorporate cluster size $|O|$ and subspace dimensionality $|S|$ directly. The score is thus the weighted sum (by parameter α) of these two properties, normalized by maximal cluster size c_{max} and maximal dimensionality d_{max}:

Definition 40. Size and dimensionality scoring:

$$score_1(o) = \sum_{o \in (O,S)} \alpha \cdot \left(\frac{|O|}{c_{max}}\right) + (1-\alpha) \cdot \left(\frac{|S|}{d_{max}}\right)$$

An object o is assigned a score for each subspace cluster it belongs to, weighted by the size and dimensionality. Objects that are not part of any subspace cluster score zero and objects in only small or very low dimensional subspace clusters score low, accurately reflecting their deviation from the prevailing patterns in the data.

Alternatively, in our second scoring function we use the measurements of density-based subspace clustering for outlier scoring. From the point of view of the density-based clustering paradigm, objects with high density are "normal". As discussed in Chapter 2, for subspace clustering this should be normalized by the expected density. Thus, the factor $\tilde{F}(o)$ by which an object o actually exceeds the expectation is an adequate weight for an object's score:

Definition 41. Density expectation scoring:

$$score_2(o) = \sum_{o \in (O,S)} \tilde{F}(o) = \sum_{o \in (O,S)} \frac{\varphi_S(o)}{E[\varphi_S]}$$

As before, objects not part of any subspace clusters are assigned a value of zero, and objects that only barely exceed the expected density are given low scores. Compared to our first scoring function this is a more fine grained scoring. By including density in the scoring it takes more information into account than the size and dimensionality. With this function, dense clustered objects seem to be more "normal" than scattered objects forming a cluster that tends to include also some outliers.

Overall, we propose two novel scoring functions which can be based on any subspace clustering result. Using high quality cluster structures one can obtain a meaningful outlier ranking. However, this work has to be seen as visionary step for outlier ranking. The result is highly depending on the quality of underlying subspace clustering. Nevertheless, we show in the following section, that it achieves meaningful results. Especially for heterogeneous high dimensional data our subspace clustering proposed in Chapter 7 provides a good bases for outlier ranking.

12.3 Experiments

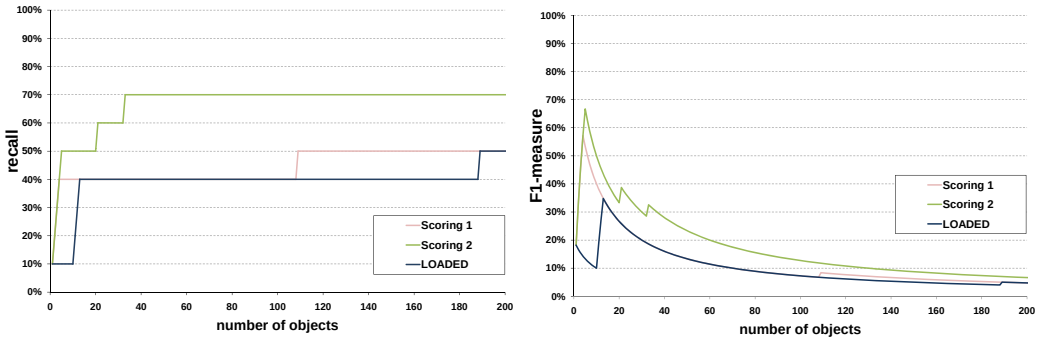
For a preliminary evaluation of our approach we constructed a test data set[1] out of a real world database containing both categorical and continuous attributes. We randomly added 10 and 100 outliers, respectively, to assess the potential of the scoring functions for outlier ranking.

We measure recall and F1-measure values known e.g. from classification [WF05]. Recall is the ratio of outliers found in the ranking by the total number of outliers in the data. It indicates to which degree the rankings are successful in detecting the hidden outliers. The F1-measure is the harmonic mean of recall and precision, i.e. it also takes the false positives into account. For 10 outliers Figure 12.2(a) and Figure 12.2(b) show recall and F1-measure values vs. different ranking sizes. Similarly, Figure 12.3(a) and Figure 12.3(b) show recall and F1-measure values for 100 hidden outliers. We evaluate both $score_1$ and $score_2$ compared to a competing approach. For $score_1$ we set $\alpha = 0.25$ by empirical evaluation. As competing approach we used the LOADED algorithm [GOP04], which is able to work on heterogeneous data. A thorough evaluation on other state-of-the-art approaches focusing only on continuous attributes is provided in the following chapters.

An ideal ranking should first find all hidden outliers and thus show both increasing recall and F1-measure until all outliers have been detected. Because we introduced outliers randomly, some of the introduced "outliers" may actually be consistent with the data distribution and not show up as outliers. Furthermore, introducing synthetic outliers in a real world database might not be the complete ground truth of hidden outliers. Real outliers hidden in the original database are not known in advance such that quality measures based only on the synthetic outliers might not result in optimal quality assessment. We tackle this additional challenge of outlier evaluation in the following chapters by using only the available real world data distributions without adding synthetic data objects. Furthermore, in the following chapters we will also

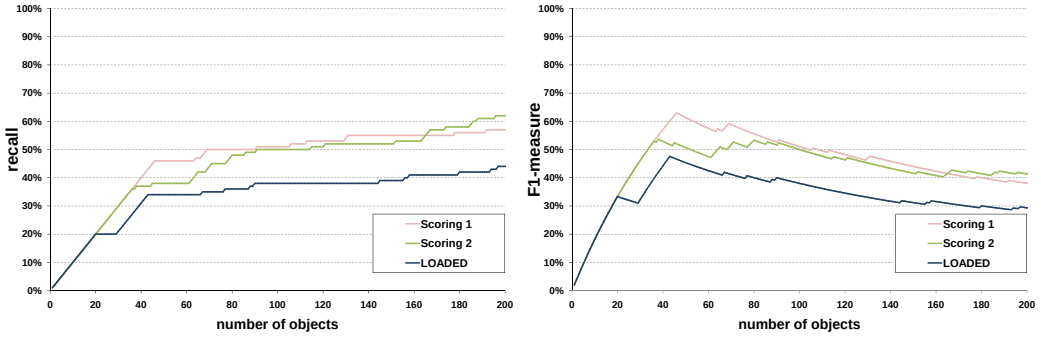
[1] data set containing 900 objects described by seven attributes. Earth quake monitoring database available at http://nsmp.wr.usgs.gov/data.html

show additional experiments on *OutRank* adding further competing approaches with experiments on a broad set of benchmark data.



(a) Recall (b) F1-measure

Figure 12.2: Outlier ranking quality for 10 hidden outliers



(a) Recall (b) F1-measure

Figure 12.3: Outlier ranking quality for 100 hidden outliers

As we can see, our subspace clustering based approach yields promising results. It shows a faster increase in both recall and F1-measure, and in sum detects more outliers on the first 200 objects than the competing algorithm. For the data set with only 10 outliers, $score_2$ shows a clearly better performance, while for 100 outliers, $score_1$ is slightly better at detecting outliers. This could be due to the fact that for more outliers, the factor by which the expected density of a subspace is exceeded better reflects the degree of deviation than the size and dimensionality of subspace clusters. For future work, it might be interesting to develop an overall optimal scoring function. Extracting more information out of our more enhanced clustering models might be considered as well as an integration of outlier ranking into the subspace clustering process.

Compared with LOADED, which assigns link scores based on different schemes for categorical and continuous attributes, our consistent outlier measure for heterogeneous data shows superior performance. Evaluating OutRank in the following chapters on a broad set of databases will show the high outlier ranking quality compared to other well established approaches specialized to continuous attributes only.

However, it is still limited due to the high dependency to the underlying subspace clustering approach.

12.4 From post-processing to direct outlier ranking in subspaces

Ranking of outliers is a useful technique for the analysis of potentially deviating objects in the data. Starting from the most unusual objects with respect to patterns in the data, the ranking can be studied up to a user specified point. The ranking should thus reflect the degree of deviation. In this work, we have studied rankings that reflect outliers hidden in subspace projections of the data. As opposed to existing approaches, our OutRank approach handles heterogeneous data, i.e. both continuous and categorical attributes, of high dimensionality. Using our enhanced models for detection of subspace clusters as pattern representatives in high dimensional data, we have developed a novel ranking for detection of outliers in any subspace. The proposed scoring functions reflect the main characteristics of the objects with respect to subspace clusters to ensure that outlier ranking reflects the degree of deviation. As based on subspace clustering result, our outlier ranking might be used as general post-processing technique. It might benefit from further enhancements of subspace clustering in the future.

Although our post-processing shows promising results, in the following we focus on a direct detection of outliers hidden in subspaces. Overcoming the drawbacks of only aggregated information provided by the subspace clustering, we directly measure density deviation in subspace projections resulting in even higher quality of outlier detection. In the following chapters we derive several outlier detection challenges. To tackle these challenges we propose an adaptive deviation that measures the degree of outlier deviation in a selection of relevant subspaces (cf. Chapter 13). Based on these subspaces we derive additional descriptive components in Chapter 14. These two novel techniques achieve not only to enhance the quality of outlier detection in subspaces but also to provide knowledge about possible outlier reasons.

Chapter 13

Adaptive outlier ranking in relevant subspaces

Compared to our previous outlier ranking method, we propose an enhanced model for outlier detection in subspace projections. As major difference we directly compute an outlier ranking without prior subspace clustering. Although we have to tackle the challenges of high dimensional data anew, this ensures more detailed analysis of outlier properties for each object. While subspace clustering has its focus on relevant subspaces for each cluster, our novel approach detects for each object relevant subspace projections in which this object shows high deviation. Furthermore, we determine the deviation locally for each object. Thus, we ensure to detect local deviation of objects w.r.t. their local neighborhoods in arbitrary subspaces.

Overall, we propose a novel outlier ranking based on the degree of deviation in relevant subspace projections. This ensures to find objects deviating in multiple subspace projections, but, it also poses new major challenges for outlier ranking. We tackle these challenges by selecting a set of relevant subspaces for outlier ranking and propose novel adaptive density and local deviation measures to determine comparable outlierness in arbitrary dimensional subspaces. In thorough experiments on real and synthetic data we show that our approach outperforms competing outlier ranking approaches especially for high dimensional data.

13.1 Motivation of outlierness in subspace projections

In general, outlier rankings provide for each object the extent of *outlierness*. However, traditional outlier rankings using outlierness measures in full space are not appropriate for high dimensional data. Due to the curse of dimensionality all objects appear to be alike in the full space so that traditional outlier rankings cannot distinguish the outlierness of objects any more.

In our work we measure outlierness (the degree of deviation) of an object in projections of the full space taking only subsets of attributes into account. Hence, we can successfully detect an outlier in a set of relevant subspace projections in which this outlier stands out from its surrounding objects. We model various behaviors of one object in a high dimensional database. An object might show high deviation com-

pared to its neighborhood in a certain subspace. In addition, the same object might cluster with some other objects in a different subspace. In contrast to this situation, the same object might not show up as an outlier in a third scattered subspace where all objects seem to be outliers. For an effective outlier ranking based on subspace projections we have to cope with all of these cases. Please note, that topics like subspace clustering [PHL04, KKZ09] and dimensionality reduction [Jol86] only seem to be related. Subspace clustering aims to detect for each cluster a set of relevant dimensions (one subspace), while we aim at detecting for each object multiple relevant subspaces in which this object deviates from its neighborhood. Global dimensionality reduction techniques like principal components analysis are not applicable as we aim at detecting multiple relevant subspaces per object while PCA provides only a single global reduction to one projection. Thus, outlier ranking in subspaces poses novel challenges not tackled by these research topics.

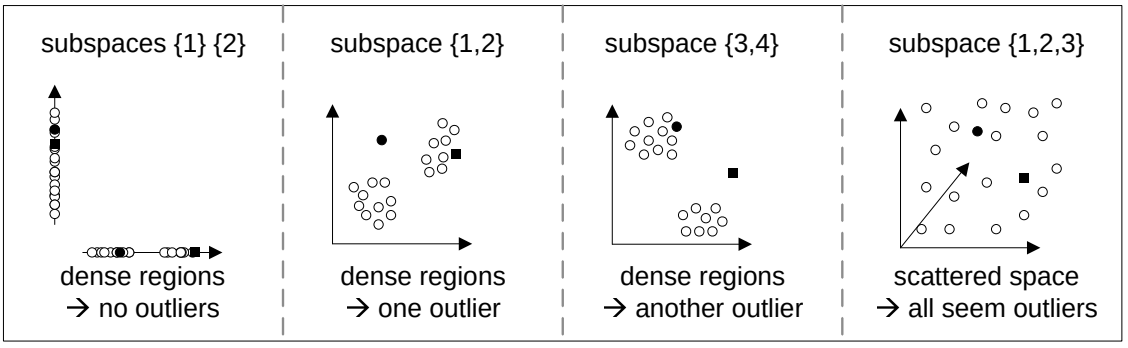


Figure 13.1: Example: Outliers in arbitrary subspaces

For illustration purposes, we have depicted several possible cases for two outliers deviating in different subspaces in Figure 13.1. In the one dimensional case $\{1\}$ and $\{2\}$ the outliers do not show up as most of the data is dense with almost no deviation. For the two dimensional cases we see that one of the outliers deviates in subspace $\{1, 2\}$ while it is clustered in subspace $\{3, 4\}$. In the three dimensional space $\{1, 2, 3\}$ both outliers disappear as objects are scattered and all objects seem to be outliers.

As illustrated, there are two key properties for outlier ranking in subspace projections:

First, outliers show up only in few projections, we call these *relevant subspaces*. In relevant subspaces most objects are clustered and outliers deviate from these clusters. In contrast, in irrelevant subspaces objects are distributed uniformly random such that all objects seem to be outliers. Thus, outlier ranking is confronted with arbitrary subspaces, while only very few are relevant and contribute to a distinction between clustered objects and really meaningful outliers.

Second, object deviation increases for increasing dimensionality. As distances between objects grow with increasing dimensionality of a subspace, objects are clustered in dense regions in one dimensional subspaces while scattered in high dimensional spaces. Thus, the deviation of objects is highly influenced by the dimensionality of

the considered subspaces. For an overall outlierness measure we have to cope with this variable density in arbitrary dimensional subspaces. In contrast to traditional outlier ranking approaches which have to adapt to local varying deviation of objects in a single space (the full data space) [BKNS00], our outlier ranking in subspaces has to consider variation of deviation in different spaces due to the varying dimensionality of considered subspace projections.

Thus, for outlier ranking in subspace projections of high dimensional data, we have to cope with two major challenges:

- Outliers appear only in relevant subspaces.
- Incomparable deviation in different subspaces.

To tackle both of these challenges, we propose a new method for outlier ranking in relevant subspace projections (OUTRES). With our novel adaptive outlierness measure we achieve to determine comparable degrees of deviation for objects in arbitrary subspaces. We define outliers to be objects highly deviating from the estimated density in their local subspace neighborhood. For a comparable degree of deviation we adapt the neighborhood and thus also the density to the actual dimensionality of each subspace. We rank all objects by aggregating the outlierness of each object for a set of relevant subspaces. As outliers are distinguished only in non-uniformly distributed projections we exclude uniformly random distributed subspaces. The overall outlier ranking is thus able to detect outliers hidden in subspaces of high dimensional data.

13.2 Comparison with related work

In general, outliers are objects that deviate from the rest of the data to a great extent. While traditional methods measure this object deviation by using all given attributes, recent approaches use only individual subsets of attributes per object. We review approaches from both categories to show their main drawbacks in contrast to our adaptive outlier ranking based on multiple relevant subspaces.

Traditional outlier mining For traditional outlier mining, different models have been proposed modeling deviation globally e.g. in distance-based [KNT00] or cluster-based [EKSX96, HXD03] outlier mining methods. However, such techniques suffer from difficulties in parametrization, as the extent of deviation is usually hard to quantify globally. This has led to outlier ranking based on the local degree of deviation for each object, e.g. as in the well established local outlier factor (LOF) approach [BKNS00]. On the other side, traditional statistical outlier detection methods consider deviation of objects only in single dimensions [BL94]. Hence, such univariate outlier detection methods are not sufficient to detect objects that deviate from their neighborhood only if one considers multiple dimensions. While these approaches have been successful in low dimensional data, high dimensional data still poses a challenge

to outlier detection as outliers appear only in relevant subspace projections. Considering object distances only in the fixed full dimensional space or taking univariate distributions in single dimensions into account, traditional outlier detection methods fail to separate outliers from the remaining data in high dimensional data sets.

A recent extension of LOF for high dimensional data proposes an angle based outlier factor (ABOF) [KSZ08]. Based on the assumption that angles are more stable than distances in high dimensional space, ABOF computes for each object an angle range to the residual objects. The angle range is used for ranking: Objects with high angle ranges are assumed to be part of clustered regions, while low ranges indicate outliers. The latter are ranked first in the produced outlier ranking. However, as ABOF is based on angles in full space it is affected by the curse of dimensionality just like LOF.

Outlier mining in subspaces In contrast to traditional outlier mining, recent approaches have been proposed that consider subspace projections for outlier ranking. The key property for all of these approaches is the appropriate choice of considered subspaces. The SOD approach considers for each object only one subspace spanned as a hyperplane by a set of reference points [KKSZ09]. Its general hypothesis states that outliers deviate within this hyperplane. However, SOD determines the outlierness of an object only in this single subspace, if objects deviate in two or more subspaces SOD is unable to distinguish between their outlier factors. Thus, the rigid definition of outliers in SOD is not appropriate for detection of outliers in multiple subspace projections.

Another approach considers multiple subspace projections for outlier ranking [LK05]. It computes the LOF outlierness for a randomly chosen subset of projections and aggregates these to an overall ranking measure. Taking arbitrary random projections into account some scattered subspaces are possibly chosen as well. Thus, these irrelevant subspaces dilute the overall quality of RPLOF. For each scattered subspace the ranking measure grows more and more alike and hinders the distinction of meaningful outliers. In preliminary experiments we compared to this approach, but, due to high runtimes we could not get any meaningful results for further evaluations.

A recent approach has been proposed for outlier detection in subspace projections as an extension of the well known subspace clustering approach CLIQUE [AGGR98]. It is based on the observation that subspace outliers have to deviate from clusters in subspace projections [AY01]. However, using the idea of fixed grid-cells for data representation, the subspace outlier detection has to rely on heuristics to overcome the tremendous amount of possibly sparse grid-cells. Furthermore, the output is a set of outliers and thus does not provide a degree of deviation for comparison with outlier ranking approaches.

Another approach is proposed by our previous work in this area. It is is based on the actual subspace clustering output. As a post-processing of subspace clustering algorithms, OutRank computes an outlier ranking by taking the number of objects and the number of relevant dimensions into account. High dimensional subspace clusters or subspace clusters covering many objects are used as indicators for low

outlier probability. However, by reducing the available information to only number of objects and dimensionality of clusters, OutRank ignores the actual deviation of each object in the considered subspace.

Overall, traditional full space outlier ranking methods simply compute object deviation in one fixed space and thus miss outliers in subspace projections. In contrast, outlier ranking in subspace projections has to cope with local deviation of objects in arbitrary dimensional projections. However, none of the proposed methods for outlier ranking in subspaces considers the highly varying density of objects due to the varying dimensionality of subspaces.

13.3 Adaptive outlier ranking in subspaces

As outliers are hidden in arbitrary projections of high dimensional databases, we propose a novel method for outlier ranking based on object deviation in subspace projections. The general idea is to measure deviation of each object in a set of relevant subspace projections. In contrast to traditional outlier ranking approaches, we consider for each object its deviation in multiple subspaces. This ensures to find objects deviating in multiple subspace projections, but, it also poses new major challenges for outlier ranking.

First, we have to cope with object deviation in different subspaces, especially we have to cope with the different dimensionality of these spaces. The varying dimensionality of subspaces induces highly varying, and thus, incomparable deviations of objects in different subspaces. To tackle this challenge, we propose a novel adaptive outlierness measure which yields comparability of object deviation in arbitrary subspaces. Hence, we may safely aggregate these adaptive outlierness values to obtain an overall outlier ranking.

Second, we have to cope with irrelevant subspace projections, like uniformly distributed subspaces. In such subspaces object deviation cannot distinguish between outliers and regular objects. Even more important, by including such irrelevant subspaces into an outlier ranking computation one loses the required contrast between the deviation of outliers and regular objects. As all objects show similar deviation in uniformly distributed subspaces, we exclude such irrelevant subspaces. In our novel outlier ranking we only include significantly non-uniformly distributed subspaces. These relevant subspaces are able to clearly distinguish between outliers and regular objects.

In this work we tackle both mentioned challenges. In Section 13.3.1, we provide basic notions before formalizing the two mentioned challenges. In the following, we first highlight how one can tackle the selection of relevant subspaces in Section 13.3.2. And second, we define an adaptive outlierness in Section 13.3.3 which measures comparable density deviation in subspaces of different dimensionality. Finally, we present the computation of the aggregated ranking value in Section 13.3.4 which specifies the overall degree of deviation for each object.

13.3.1 Formalization of challenges for outlier ranking

The general aim of outlier ranking is to provide a sorting of all objects o given in a database DB. Technically, one ranks according to the *degree of deviation* measured by a ranking function $r : DB \to \mathbb{R}$. The ranking function provides a real valued measure of the objects' outlierness. Ranking functions can be defined arbitrarily based on the object's features $o = (o_1, \ldots, o_d)$. In contrast to traditional approaches that measure the degree of deviation in the full d-dimensional space $D = \{1, \ldots, d\}$, we measure deviation in subspace projections $S \subseteq D$. Thus, we ensure to find outliers hidden in arbitrary subspace projections.

As a density-based approach, OUTRES measures the degree of deviation according to the density $den(o, S)$ of an object in subspace S. We observe that low density values $den(o, S)$ indicate that o is most probably an outlier in subspace S. Hence we derive our novel adaptive outlierness measure $out(o, S)$ based on the object density. We will propose our outlierness function $out(o, S)$ in Section 13.3.3, where we will also give details about the underlying adaptive density $den(o, S)$ for a comparable outlierness measurement. Please keep in mind that we set low values in $out(o, S)$ for highly deviating objects as we sort our ranking in ascending order. Thus our overall ranking function is computed as described in Definition 42 aggregating outlierness over all relevant subspaces.

The general challenge for outlier ranking approaches is to provide a meaningful ranking function which achieves to distinguish between an outlier object o and a regular object p by providing a clear distinction: $r(o) \ll r(p)$. However, traditional outlier ranking functions fail in high dimensional data as they provide for all objects very similar ranking values $r(o) \approx r(p)\ \forall o, p \in DB$. This can be explained by the curse of dimensionality, as traditional ranking functions consider all dimensions for distance computation $dist_D(o, p)$. In our case we use the Euclidean distance $dist_D(o, p) = \sqrt{\sum_{i \in D}(o_i - p_i)^2}$, where distances between $o \in DB$ and any other objects grow more and more alike with increasing dimensionality $|D| \to \infty$:

$$\lim_{|D| \to \infty} \frac{\max_{p \in DB} dist_D(o, p) - \min_{p \in DB} dist_D(o, p)}{\min_{p \in DB} dist_D(o, p)} \to 0$$

As consequence, ranking values based on these full space distances become meaningless:

$$\lim_{|D| \to \infty} r(o) - r(p) \to 0 \Rightarrow r(o) \approx r(p)\ \forall o, p \in DB$$

Although outliers do not show up in full space, they show high deviation in subspace projections. Thus, we cope with the curse of dimensionality by considering the deviation of each object in a set of relevant subspace projections RS. We measure the outlierness $out(o, S)$ by restricting distance functions $dist_S(o, p)$ to the subspace dimensions in S. The overall ranking value $r(o)$ of an object o is then simply computed by the product of its outlierness in each relevant subspace $S \in RS$ as given in Definition 42. As aggregation function we considered also the sum or the maximum over all outlierness measures. However, in preliminary experiments the sum has shown only low contrast between outliers and regular objects. By summing

up all measures, one observes a high influence of each individual measure while using the product one highlights outlier objects with significantly low outlierness values. On the other side, the maximum takes only the most significant outlierness value. Thus, it provides only ranking information of one subspace projection. Overall, using the product in our aggregated ranking we observe best ranking results.

Definition 42. Subspace Ranking Function
The overall ranking value $r(o)$ of an object $o \in DB$ w.r.t. a set of relevant subspaces RS and an outlierness measure $out(o, S)$ is defined as:

$$r(o) = \prod_{S \in RS} out(o, S)$$

As we measure the outlierness in subspace S, $out(o, S)$ only considers the dimensions in S for distance computations $dist_S(o, p)$ between any objects $o, p \in DB$.

While traditional ranking functions consider the outlierness of an object only in the full space D, we aim at considering outlierness in a set of subspaces $RS \subseteq \mathcal{P}(D)$ out of the powerset of possible subspace projections. This is meaningful for high dimensional data as outliers are hidden in multiple subspace projections. Thus, in contrast to the degree of deviation in full space, our outlier ranking can distinguish between outliers and regular objects by using a set of relevant subspace projections. However, two novel challenges arise:

- How to choose the set of relevant subspaces RS for meaningful outlier ranking
- How to achieve comparable outlierness values $out(o, S)$ over multiple subspaces $S \in RS$.

We formally derive these two challenges, before presenting our solution in the following. Both challenges can be derived from the curse of dimensionality. While objects are dense in low dimensional spaces, for higher dimensional spaces they diverge until they form a uniformly distributed scattered space. In Figure 13.2 we show a box-plot for the varying distribution of density in various dimensions as a toy example. For 1d subspaces all objects are dense and show almost no deviation, while 3d and 4d spaces are scattered with overall low density. The hidden outlier shows up only in some of the 2d subspace S. In the depicted 2d spaces, we observe that outliers show up as objects deviating from clustered regions in their local neighborhood. Outliers disappear if no clusters exist or if they form a cluster. In both cases they show very similar density and almost no deviation to their local neighborhood.

Our key hypothesis is that outliers can be distinguished in non-uniformly distributed subspaces. However, objects might be outliers in multiple subspaces, thus, a meaningful outlierness measure has to be comparable over different subspaces. Formally, for a meaningful outlier ranking we have to tackle the following challenges:

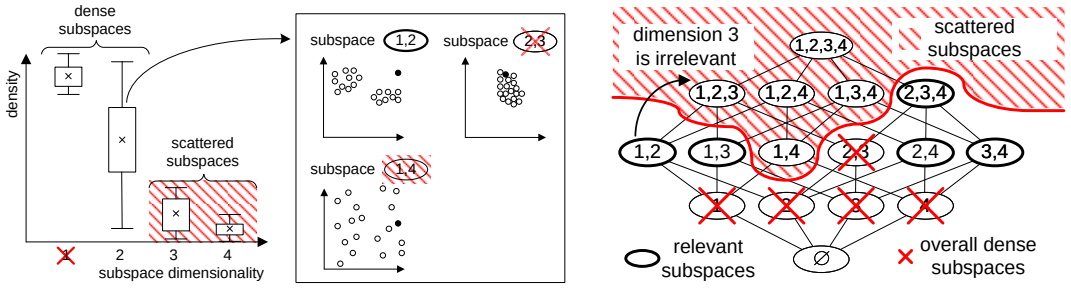


Figure 13.2: Relevant subspace projections for outlier mining

Outlier Detection Challenge 1. Comparability of Outlierness
Outlierness measures are not comparable over multiple subspaces if: For subspace $S, T \subseteq D$ *with* $T \subset S$

$$
\begin{aligned}
& \textit{due to curse of dimensionality} \\
\Rightarrow\ & \forall p \in DB : dist_S(o,p) \geq dist_T(o,p) \\
\Rightarrow\ & den(o,S) \leq den(o,T) \\
\Rightarrow\ & out(o,S) \leq out(o,T) \\
\Rightarrow\ & \textit{outlierness is biased w.r.t. dimensionality}
\end{aligned}
$$

As density drops for increasing dimensionality, outlierness measures based on density in subspace projections are biased w.r.t. the dimensionality of the considered subspaces. Thus, overall aggregation (cf. Def. 42) of outlierness is hindered by incomparable measures. With such an incomparable measure, high dimensional spaces would dominate the ranking value and outliers in low dimensional projections could not show up in the overall ranking. Thus, as we take multiple subspaces into account, we have to provide an adaptive outlierness measure with comparable outlierness in arbitrary subspace projections to achieve a fair ranking of objects in any dimensional subspaces.

Outlier Detection Challenge 2. Relevance of Subspaces
A subspace S *hinders the distinction of outliers if:*

$$
\begin{aligned}
& S \textit{ is distributed uniformly random} \\
\Rightarrow\ & \forall o,p,q \in DB : dist_S(o,q) \approx dist_S(p,q) \\
\Rightarrow\ & \forall o,p \in DB : den(o,S) \approx den(p,S) \\
\Rightarrow\ & \forall o,p \in DB : out(o,S) \approx out(p,S) \\
\Rightarrow\ & \textit{distinction of outliers is hindered}
\end{aligned}
$$

Obviously the full space D *is such an irrelevant subspace for increasing dimensionality* $|D| \to \infty$

With decreasing density, one reaches subspaces with uniformly distributed objects where outliers do not show up. Including such an irrelevant subspace projection S into a ranking function yields very similar ranking values for all objects. Thus, our key property for the set of relevant subspaces is to exclude subspaces which are distributed uniformly random.

13.3.2 Selection of relevant subspaces

First, we propose the selection of a set of *relevant subspaces* RS that can distinguish between objects according to their degree of deviation. This is clearly not possible in scattered subspaces that show uniformly random distributed data. Thus, we propose to exclude such scattered subspaces from outlier ranking by testing the underlying distribution of each subspace. Our test is based on a statistical significance test aiming at reducing the probability that a uniformly distributed subspaces passes into the set of relevant subspaces. As uniformly distributed subspaces hinder the detection of meaningful outliers, we ensure with a given significance level α that uniformly distributed subspaces are only included into the ranking with a very low probability of less than α. Further details on our significance test are provided in Chapter 14.

To illustrate the effects of relevant subspace selection, we depict a subspace lattice with all possible subspaces of a 4d data space in Figure 13.2. Starting considering 1d projections first, typically these low dimensional projects of high dimensional data are uniformly dense. The whole database seems to be one dense region. Furthermore, outliers in 1d projections could be easily detected as pre-processing. By including more and more dimensions, due to correlations of the data, the database diverts in multiple dense regions. For our outlier ranking we only take the outlierness of objects in these relevant subspaces into account. Adding even more dimensions the subspaces become scattered like the full space. All objects seem to be outliers. Incrementally using the statistical test for each dimension we detect the irrelevant dimensions and stop further processing of higher dimensional subspaces.

Thus, we define the relevant subspaces RS in Definition 43 to be the set of subspaces that are significantly non-uniformly distributed.

Definition 43. Set of Relevant Subspaces
The set of relevant subspaces contains subspaces that are significantly non-uniformly distributed:

$$RS = \{S \in \mathcal{P}(D) \mid S \text{ passes significance test } \}$$

Only these subspaces are considered for our outlier ranking (cf. Def. 42).

For good ranking quality, we have to ensure that uniformly distributed subspaces are only included in very rare cases by setting a low value for the significance level α. Thus, we ensure a low probability of uniformly random distributed subspaces in our outlier ranking. We will show the influence of the α parameter for the overall outlier ranking quality in Section 13.4.

Although we have excluded the irrelevant dimensions a major challenge remains. Subspaces in RS have arbitrary dimensionality and show highly varying density and deviation values.

13.3.3 Adaptive outlierness in subspaces

For a meaningful outlier ranking based on outlierness in multiple subspace projections the definition of $out(o, S)$ has to provide an adaptive outlierness measure as the overall ranking combines object properties out of very different subspaces $S \in RS$.

We propose such an *adaptive outlierness* measure by defining an *adaptive density* and a *local deviation* for each object.

As formalized in Challenge 1, measuring density in multiple subspaces of arbitrary dimensionalities leads to a challenging task, namely the strong dependence of densities on the dimensionality. In order to find meaningful outliers that deviate considerably from the underlying local density distribution, we propose an adaptive density computation so that outlierness of objects in subspaces of different dimensionalities becomes comparable and automatically adapts to their dimensionalities. In contrast to our previous approaches proposing unbiased density-based clustering in Chapter 2, here we have to consider both unbiased density and adaptive deviation for our outlierness measurement.

For our outlier ranking based on deviations of density we first compute the density for each object and compare it with the local (average) density in a relevant subspace. By that, our approach is able to detect objects highly deviating from the residual data in a relevant subspace, i.e., objects having exceptionally low densities. While our adaptive density ensures comparability over multiple subspaces, our local deviation ensures meaningful outlierness values inside one subspace. Hence, in addition to the adaptive density, we ensure to highlight an outlier with very lower density compared to its local neighborhood in the considered subspace.

In order to derive our adaptive outlierness, we first introduce our adaptive density for arbitrary subspace dimensionalities. Second, we derive our local deviation by comparing density of an object to the average density in its neighborhood.

Adaptive object density As density drops with increasing dimensionality, we have to ensure that our density measure adapts to the expected decreasing density. Our flexible outlier model OUTRES could be instantiated with any adaptive density measure which is able to adapt to the increasing dimensionality of subspace projections. As a good instantiation of our model, we propose our adaptive density measure scaling the ε-neighborhood w.r.t. the dimensionality of S as used also for subspace clustering approaches in Chapter 3 and 4. We base on well established density estimation techniques [Sil86]. And for comparable outlierness over arbitrary subspaces, we propose to adapt the density by a variable kernel bandwidth as ε-neighborhood. For a fixed dimensionality d, optimal $\varepsilon_{optimal}(d)$ is given in [Sil86] by the following formula:

$$\varepsilon_{optimal}(d) = \left(\frac{8 \cdot \Gamma(\frac{d}{2}+1)}{\pi^{\frac{d}{2}}} \cdot (d+4) \cdot (2\sqrt{\pi})^d \right) \cdot n^{\frac{-1}{d+4}}$$

where $n = |DB|$ is the size of the database and Γ stands for the gamma function with $\Gamma(n+1) = n \cdot \Gamma(n)$, $\Gamma(1) = 1$, $\Gamma(1/2) = \sqrt{\pi}$.

Assuming that n is fixed in a static database, we observe $\varepsilon_{optimal}(d)$ to be a monotonically increasing function. So intuitively, for increasing dimensionality the neighborhood of each object is increased as well in order to maintain optimal density estimates, while the data space is becoming sparse. For comparable outlierness we use $\varepsilon_{optimal}(d)$ to adapt density estimation in arbitrary subspaces. By the parameter ε we allow the user to quantify a notion of locality and adjust this value to application

dependent properties. In contrast to adaptive density in Chapter 3, we start scaling from 2d spaces as we assume that 1d outliers can be filtered out by univariate approaches. Formally, the ε-neighborhood for arbitrary dimensionalities is given by Definition 44:

Definition 44. Adaptive ε-neighborhood
For a subspaces dimensionality d, $d \geq 2$, the adaptive neighborhood $\varepsilon(d)$ is defined by

$$\varepsilon(d) = \varepsilon \cdot \frac{\varepsilon_{optimal}(d)}{\varepsilon_{optimal}(2)}$$

Thus, we simply scale the given starting neighborhood ε from 2d space up to full data space and use these value for density estimation. Consequently, our automatic adaptation ensures comparable density estimates for arbitrary dimensional subspaces.

Local object deviation Having such a comparable density estimation, an outlier can be detected as an object showing significantly low density. As we aim at a local outlierness we measure deviation based on an adaptive threshold. As first filter step we select only objects with significantly low density $den(o, S) < \mu - 2 \cdot \sigma$. From statistical observations, only very rare objects deviate more than two standard deviations from the mean value (cf. Chebyshev's inequality [HLP88]). As statistical probability for such objects is low (e.g. for normal distributed data it is less than 2.1%), their outlierness has to be high. Using mean μ and standard deviation σ of the estimated (local) density we ensure to be adaptive to varying density. Object deviation is then defined by:

Definition 45. Object deviation
The deviation of an object o with respect to mean and standard deviation of the estimated density:

$$dev(o, S) = \frac{\mu - den(o, S)}{2 \cdot \sigma}$$

Adaptive Outlierness Overall the outlierness of an object o has to fulfill two major requirements. First, it has to be adaptive to arbitrary dimensional subspaces. Thus, based on our adaptive object density we propose an adaptive outlierness which is comparable for different dimensionalities. Second, our adaptive outlierness has to cope with object deviation considering statistically deviation from the mean value.

Thus, our novel outlierness incorporates both aspects derived by density and the deviation of each object: low density and high deviation are both indicates for high outlierness. Highly deviating objects show up by $dev(o, S) \geq 1$ as density is significantly low compared to mean and standard deviation. Thus, the outlierness measure is defined as follows:

Definition 46. Local Outlierness
The outlierness of an object o in subspace S is derived by its density and its deviation

in this subspace:

$$out(o,S) = \begin{cases} \frac{den(o,S)}{dev(o,S)} & , \text{if } dev(o,S) \geq 1 \\ 1 & , \text{else.} \end{cases}$$

For technical reasons we define $out(o,S)$ in the range $(0 \ldots 1]$ with low values indicating highly deviating objects. For objects with lower deviation we assign no outlierness resulting in bounding $out(o,S)$ to a maximal value of 1. Thus, outlierness can be easily aggregated for arbitrary subspaces. Overall we cope with the different behaviors of objects in different subspaces: Scattered irrelevant subspaces are excluded by our relevance testing (cf. Sec. 13.3.2). Objects in a dense subspace S result only in high density and almost no deviation such that we set $out(o,S) = 1$ they do not affect the ranking value. Only if objects show up with low density or high deviation in a relevant subspace they contribute to the overall ranking value.

13.3.4 Computations of outlier ranking

A naive computation of our outlier ranking would have to compute relevance of arbitrary subspaces and the density of each object in these subspaces. This is computationally impractical for two reasons. First, there are exponentially many of these subspaces which have to be analyzed. Second, density estimation is a computationally expensive task with quadratic runtime with respect to the number of objects. Thus, resulting in similar efficiency challenges as discussed in Part II of this thesis.

For efficient computation of our outlier ranking we base our algorithm on two commonly used techniques. For an efficient processing of subspace we use a bottom-up algorithm, starting at 1d projections and recursively mining higher dimensional projections. Such processing has been successfully applied for efficient subspace clustering as described in Chapter 5. We extend this recursive processing of subspaces for our outlier ranking by pruning subspace projections according to our relevance test (cf. Section 13.3.2). Having reached a sparse subspace with uniformly distributed data we stop processing, as data is scattered even more in higher dimensional projections. This ensures to exclude all irrelevant subspace projections, which would hinder the distinction of outliers in our ranking. Overall, as depicted in Figure 13.2 we detect only a small set of relevant subspace projections in which we have to perform density estimation achieving an efficient processing even for high dimensional spaces.

As we keep the focus on the quality of our outlier ranking model we skip further descriptions of the efficient density computation. By using efficient techniques for processing arbitrary subspace regions in high dimensional databases as proposed in Chapter 5, we achieve OUTRES to be a practically applicable approach. We have performed various experiments on synthetic and real world data, which showed competing runtimes with state of the art outlier ranking approaches like LOF (taking only the full space into account). However, even more important is that OUTRES achieves high outlier ranking quality which is shown in the following section.

13.4 Experiments

We demonstrate the quality of our OUTRES approach on both synthetic and real world data. We compare OUTRES to the well established LOF [BKNS00] and its recent extensions ABOF [KSZ08] and SOD [KKSZ09] for high dimensional data. Furthermore, we compare against our own approach OutRank, proposed in the previous chapter.

13.4.1 Evaluation measures

For comparison of these outlier ranking methods we use three different quality measures. We measure true positive (TPR) and false positive (FPR) rates visualized in the well established ROC plot known e.g. from classification [WF05]. Both of these measures are useful to derive if a ranking detects a high ratio of correct detected outliers (TPR) while providing only few non-outlier as detected outliers (FPR). However, they only take the ratio of detected outliers and non-outliers into account ignoring more or less the positioning of the objects in the ranking. Thus, we additionally evaluate the results with a ranking coefficient based on Spearmans Ranking Coefficient [Spe87]. In contrast to the ROC plot, ranking coefficients take also the ranking positions of detected outliers into account. This leads to a more fine grained quality measure.

To illustrate the quality of the rankings we use the quality measures for the top-k ranked objects (cf. Definition 47). The TPR measure is simply the fraction of found true outliers in the first k objects $\textit{found true outliers}(R, k) = \{o_{r_1} \dots o_{r_k}\} \cap DB_{\textit{hidden outliers}}$ compared to the set of hidden outliers $DB_{\textit{hidden outliers}}$ in the database DB. Analogue, FPR is the fraction of found non-outliers in the first k objects $\textit{found false outliers}(R, k) = \{o_{r_1} \dots o_{r_k}\} \setminus \textit{found true outliers}(R, k)$ compared to the overall set of non-outlier objects in the database.

Definition 47. TPR **and** FPR **measures**
The true positive rate for the first k objects of a ranking $R = \{o_{r_1} \dots o_{r_n}\}$ *is defined as:*

$$TPR(R, k) = \frac{|\textit{found true outliers}(R, k)|}{|DB_{\textit{hidden outliers}}|}$$

The false positive rate is defined as:

$$FPR(R, k) = \frac{|\textit{found false outliers}(R, k)|}{|DB_{\textit{hidden non-outliers}}|}$$

More detailed measures can be derived by ranking coefficients [Spe87]. Spearmans Ranking Coefficient $SRC(R_1, R_2)$ computes the correlation of two given rankings R_1 and R_2. We use SRC to measure the quality for one ranking by comparing it with the optimal ranking R_{best}, ranking all outliers first. Furthermore, we normalize with the ranking coefficient for the worst ranking R_{worst} having all outliers in the last positions. We define outlier ranking coefficient $ORC(R, k)$ for the first k objects in ranking R as given in Definition 48.

Definition 48. Ranking coefficient measure
The outlier ranking coefficient for the first k objects of a ranking $R = \{o_{r_1} \dots o_{r_n}\}$ *is defined as:*

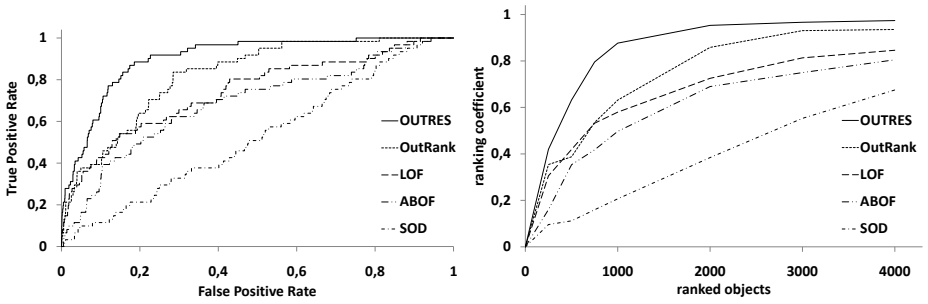
$$ORC(R,k) = \frac{SRC(\{o_{r_1} \dots o_{r_k}\}, R_{best})}{SRC(R_{worst}, R_{best})}$$

For both measures the optimal ranking results in $ORC(R_{optimal}, k) = 1$ and $TPR(R_{optimal}, k) = 1 \wedge FPR(R_{optimal}, k) = 0$ for $k = |DB|_{outliers}$. For non-optimal rankings $TPR = 1$ is reached for larger k with $FPR \gg 0$, while the ORC measure does not reach the maximal value of 1 at all for non-optimal rankings. Thus, the ORC measure is more appropriate for evaluation of outlier rankings. By taking the actual positing of objects into account, ORC is able to distinguish between two rankings having found the same amount of outliers in the first k positions. In such a case, TPR and FPR show same results as they only consider the object ratio and cannot distinguish between these two rankings. Taking also positioning information into account ORC shows more fine grained differences in rankings. Especially, one can compare ranking quality by taking the overall $ORC(R, |DB|)$ for comparison. Thus, after showing all three measures in the first experiment we use only the ranking coefficient measure for comparison in the following experiments.

13.4.2 Evaluation on synthetic data

For scalability experiments, we generate density-based clusters in arbitrary subspaces as in the previous chapters. In addition, our generator adds outliers deviating from one of these subspace clusters. As there are no global patterns hidden in the high dimensional data space, the hidden outliers do not appear in full space. In our first experiment, we evaluate the quality of the competing approaches on a 15-dimensional synthetic data set with 4765 objects represented by four subspace clusters and 61 hidden outliers. Figure 13.3(a) illustrates the quality with respect to ROC plot. We observe that all approaches show high increase in true positive rates of detected outliers with only very few false positive. However, all hidden outliers ($TPR = 1$) are found after thousands of considered objects, indicated by $FPR \gg 0$. Our novel OUTRES shows best performance compared to LOF, ABOF, SOD and OutRank, as it archives to detect more hidden outliers within the first ranked objects showing both higher TPR and lower FPR than the competitors. Comparing ROC plot and ranking coefficient in Figure 13.3 for the same experiment, we observe that OUTRES outperforms all competing approaches in both quality measures. For ORC measure it always shows higher correlation with the optimal ranking than any other method. Especially, the overall measure $ORC(R, |DB|)$ of OUTRES shows the highest correlation with the optimal ranking.

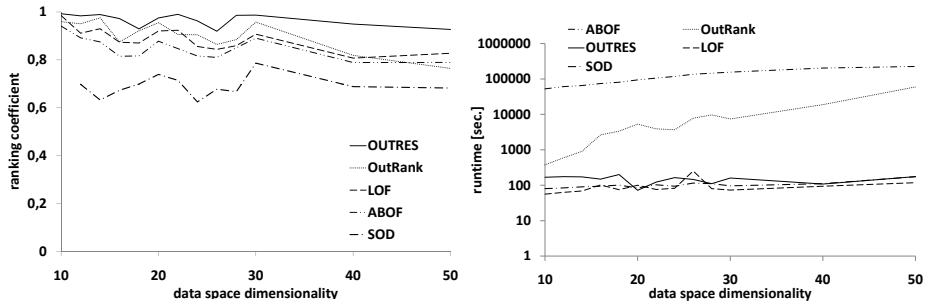
In our second experiment, we evaluate the scalability of outlier ranking with respect to the dimensionality of the data set, by comparing $ORC(R, |DB|)$ for different approaches. As outlier ranking in subspace projections aims to detect outliers in high dimensional data, scalability w.r.t. dimensionality is crucial. We varied the overall dimensionality of the data from 10 up to 50 dimensions, while keeping number of hidden subspace clusters, hidden outliers and database size constant, as in the



(a) ROC plot (b) ranking coefficient

Figure 13.3: Ranking quality on synthetic data

previous experiment. Figure 13.4(a) shows decreasing quality with increasing data dimensionality. As we add more and more dimensions, hidden outliers disappear in the overall scattered objects. However, as OUTRES investigates only relevant subspace projections it is less affected by high dimensional data. It outperforms all competing approaches.



(a) ranking quality (b) runtime

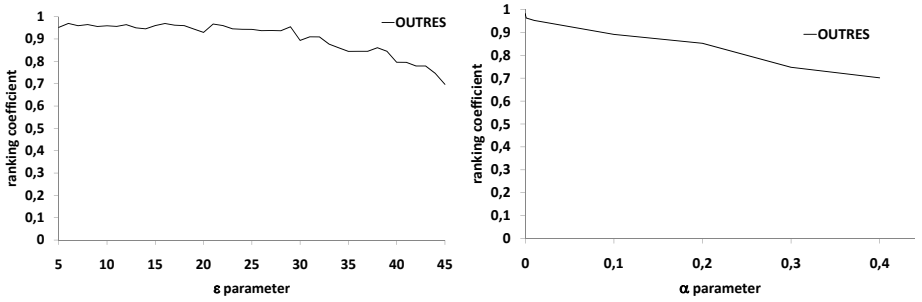
Figure 13.4: Scalability w.r.t. total number of dimensions

In Figure 13.4(b) we compare runtimes for increasing dimensionality. OUTRES, LOF and SOD significantly outperform the other approaches in terms of runtime (please note the logarithmic scale). However, in contrast to LOF and SOD which show low ranking quality, OUTRES achieves to perform both efficient outlier ranking and a high outlier ranking quality. Further experiments have shown that scalability w.r.t. database size has less impact on both quality and runtime. Thus, we skip the presentation of these experiments.

13.4.3 Parameterization

For the two main parameters α and ε we show the robustness of the ranking quality of OUTRES. On the synthetic data set from previous experiment we varied the neighborhood parameter ε from 5 to 45 (data ranges from 0 to 100). As depicted in Figure 13.5(a), OUTRES shows a quite robust ranking quality only slightly decreasing for high ε values. By increasing the neighborhood around each object density is increasing for all objects. Especially for outliers, density is becoming similar to

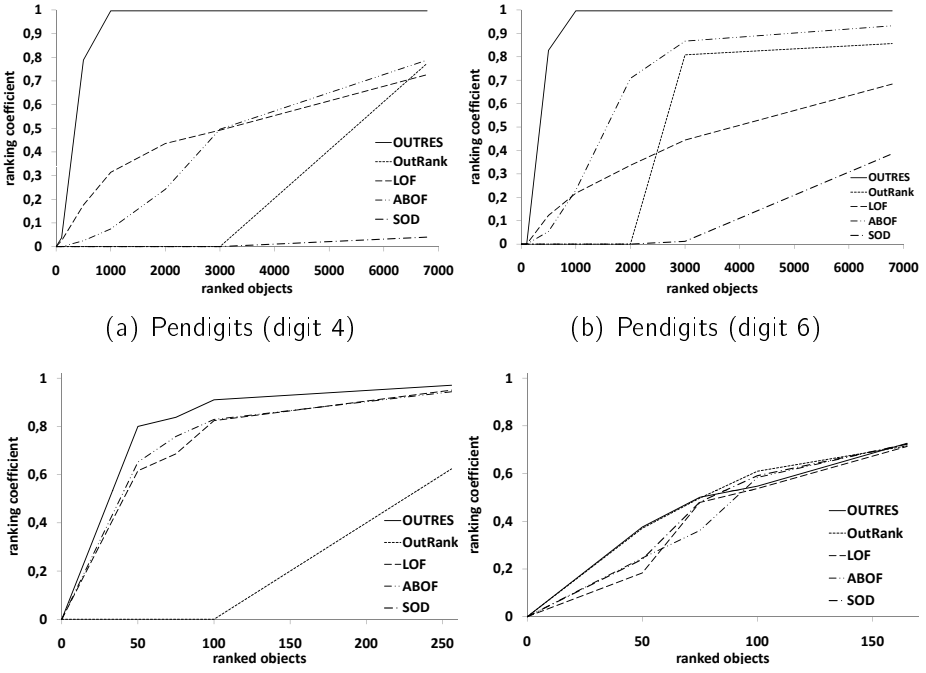
clustered objects. Overall we achieve a robust approach w.r.t. ε due to our automatic adaptation of the neighborhood range for the arbitrary subspace projections considered in OUTRES (cf. Def. 44). As default setting of ε in our experiments we use $\varepsilon = 15$ showing best results.



(a) Variation of ε (b) Variation of α

Figure 13.5: Robustness of OUTRES w.r.t parameters

For the second parameter α we observe similar effects. As depicted in Figure 13.5(b) we observe best ranking quality for low α settings. For higher α settings, OUTRES accepts more and more uniform distributed subspaces as relevant subspaces for outlier ranking. As one cannot distinguish between outliers and regular objects in these scattered subspaces, the overall ranking quality decreases. Keeping a low α setting (default $\alpha = 0.01$), thus, ensures to measure outlierness of objects only in relevant subspace projections.



(a) Pendigits (digit 4) (b) Pendigits (digit 6)

(c) Ionosphere (d) Breast Cancer

Figure 13.6: Ranking quality on real world data

13.4.4 Evaluation on real world data

We analyzed the quality of outlier ranking on three real world data sets (Ionosphere, Breast Cancer and Pendigits) from the UCI repository [AN07]. To measure outlier ranking quality on real world data where no ground truth on the hidden outliers is known, we used one of the class labels as ground truth for hidden outliers. However, as all of these data sets are designed for classification tasks they provide for each class sufficient many objects for a training phase. As outliers are assumed to be rare observations in a data set, we reduced one class by randomly sampling to 10% of the database size creating an artificial outlier class. For Ionospere and Breast Cancer we used the smaller of the two given classes as outliers, while for Pendigits we picked each digit as outlier class by extracting 10 different data sets. By using this outlier class as ground truth our evaluation shows that an outlier ranking approach successfully detects these very rare hidden observations in the data set.

In Figure 13.6 we show top-k ranking coefficients $ORC(R, k)$ for the competing approaches. For all data sets we observe a high ranking quality of OUTRES, outperforming competing approaches by detecting outliers as top ranked objects. Especially for Pendigits, we observe always best ranking quality for OUTRES, while the competing approaches show varying ranking quality.

13.5 Enhancements for outlier detection in subspaces

In this work, we proposed a novel outlier ranking for objects deviating in subspace projections of high dimensional data. The OUTRES approach computes local density deviation of objects by looking at subspaces of the high dimensional data. For comparable outlierness measures in different subspaces, we derive an adaptive density measure which automatically adapts to the subspace dimensionality. OUTRES computes an overall outlier ranking by aggregating outlierness of objects in relevant subspaces. Relevance of subspaces is measured by statistical significance tests. Thus, only relevant subspaces that are not distributed uniformly random are used for our outlier ranking. Our thorough evaluation on both synthetic and real world data shows that OUTRES outperforms competing outlier ranking approaches. Especially for high dimensional data where objects are scattered in full space, OUTRES achieves to detect outliers hidden in subspace projections.

Our outlier ranking shows significant enhancement in the detection of outliers hidden in subspace projections, the selection of these relevant subspaces provides additional potential for knowledge extraction. These subspaces can be seen as witnesses for the objects outlier properties as these attributes provide the reasons why an objects seems to be an outlier. Based on this observation we propose novel descriptive components in the following chapter. These components enrich the outlier ranking and provide for each object additional knowledge about the reasons of being an outlier.

Chapter 14

Descriptive components for outlier detection in subspaces

Outlier detection is an important mining task to detect highly deviating objects. For knowledge extraction further descriptive information is required in addition to the pure detection of outliers. However, recent outlier mining approaches focus only on the detection and miss to provide reasons why an object should be considered as an outlier.

In this chapter, we propose a descriptive outlier ranking. For each object we detect subspace projections describing the reasons for its outlier properties. We enhance outlier ranking by considering object deviation only in these relevant subspaces. In thorough experiments we show that our approach outperforms competing outlier ranking approaches in detection of outliers, and provides additional descriptive information about the outlier properties.

14.1 Motivation and comparison with related work

In general, the task of knowledge discovery in databases is twofold. On the one side data mining methods try to *detect* meaningful patterns, while on the other side knowledge is extracted out of the data by *providing descriptions* of these patterns. Especially, for the unsupervised outlier mining task, knowledge discovery does not end with the detection of the highly deviating objects. In applications like fraud detection, health surveillance, customer segmentation or sensor monitoring, one is interested in additional descriptions about the reasons why an object seems outlying. For example in health surveillance, a young patient might be considered as outlier due to high risk of dehydration. While dehydration is quite normal for elderly people, it is quite rare for young persons. By looking at a set of measured attributes, one might detect this outlying patient showing high deviation from the residual patients in the attributes "age" and "skin humidity". However, not only the detection of such a high risk patient but also the underlying outlier properties are important. Providing information about the high deviation in age and skin humidity while showing normal measurements in all other attributes assists health professionals in verifying this automatically detected outlier. Thus, an obvious aim for outlier detection methods

is to provide additional information about outlier properties like in which subsets of the attributes and to which extend a deviation from the regular objects (*degree of deviation*) can be observed.

Traditional methods use all available attributes (full data space) to determine the degree of deviation. In contrast, we focus on subsets of the given attributes as for many recent applications outliers show up only in *subspace projections*. In our health surveillance example, one measures many attributes to detect arbitrary diseases while each outlier is present only in a combination of few attributes. In Figure 14.1, we depict two outliers in three possible subspace projections for our toy example. As illustrated the hidden outliers show up only in specific projections while are hidden in other projections. For each outlier these specific projections can be seen as witnesses for its outlier properties.

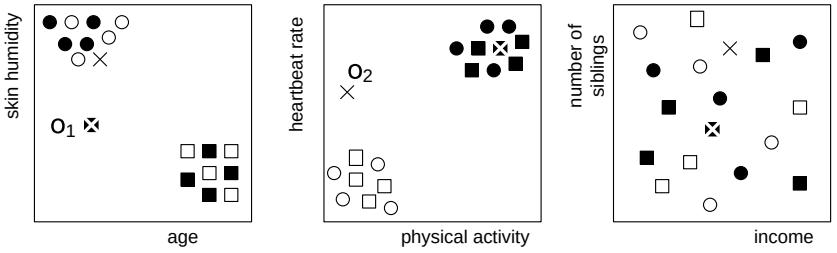


Figure 14.1: Toy example (health surveillance)

Thus, one has to detect these relevant subspace projections in which an outlier shows up by a high degree of deviation compared to its local neighborhood. This selection of subspaces is highly challenging, as too many attributes (e.g. the full data space) provide no outlying properties by showing high deviation for all objects. As this property hinders a correct outlier detection, our key idea is to measure object deviation only in subspace projections where the object's neighborhood is not uniformly distributed. Thus, we select these subspaces locally for each object. This selection of subspaces might seem similar to other mining approaches which are hindered by too many given attributes. However, our solution can be clearly distinguished from typical solutions like dimensionality reduction. We focus on a local selection of subspaces for each object while dimensionality reduction removes attributes globally for all objects. Global dimensionality reduction is not applicable in our case. Compared to typical data sets which are called high dimensional, we have given only 10-50 attributes, but each of them is relevant for our task and can not be globally removed. Each of the given attributes is required locally for the detection of at least one outlier. In our example, detection of o_1 as an outlier requires only two attributes, but none of the other attributes can be removed. Even the scattered attribute "income" is useful for detection of outliers in wealth related diseases (not depicted in our small example).

There have been proposed some outlier mining approaches focusing on subspace projections. Their key idea is that outliers show high deviation from clustered objects in some subspaces. However, they differ in their choice of subspaces, some approaches [KKSZ09, FMW08], are limited to choice of one single subspace and thus unable to detect an outlier which is hidden in multiple subspaces. Both other approaches select multiple subspaces with more or less meaningful heuristics. Ran-

dom choice of subspaces [LK05], does not guarantee high quality results as it might choose only irrelevant subspaces showing no outliers at all. In contrast our OutRank approach (cf. Chapter 12), is based on a post-processing of subspace clusters. Thus, the outlier ranking is highly depending on the outlier awareness of the underlying clustering technique. In all of these approaches, we observe that a meaningful selection of subspaces is crucial for outlier detection. Thus, in this work, we propose a local selection of relevant subspaces providing individual outlier properties for each object.

Overall, there have been different outlier detection paradigms proposed in the literature. However, all of the outlier mining approaches reviewed in Section 13.2 have a major drawbacks. They do not provide information about the reasons why an object seems to be an outlier. In general, the proposed outlier ranking approaches have their focus on outlier *detection*, they provide only the ranking values as descriptive components. Thus they are limited to providing a sorted list of most probable outliers, without giving explanations why an object seems to be an outlier. In contrast, descriptive outlier methods have their focus on *providing descriptions* about the outliers [KN99, AFP09]. However, these approaches assume that the outliers are given in advance. Thus, they provide only a post-processing to a given outlier detection. They might miss some important properties as they are not aware of the underlying outlier detection process.

We propose *DescOut*, a descriptive outlier ranking considering object deviation in subspace projections. Our key idea is to detect for each object a set of subspaces by statistical methods. We tackle both challenges: First, detecting outliers by sorting objects according to their deviation in subspaces. And second, providing descriptive components as reasons why an object seems to be outlying. Overall, the main contributions of *DescOut* are:

- Enhanced outlier ranking by choice of relevant subspace projections.
- The relevant subspaces of an outlier and a comparison to its local neighborhood as witnesses for the object's outlier properties.

Considering both, *DescOut* is the first knowledge extraction approach for outlier mining with a high quality outlier ranking considering subspace projections and additional outlier description components.

14.2 Descriptive outlier ranking

In contrast to existing approaches which perform both mentioned tasks separately, our aim is to perform outlier detection and outlier description in one approach. This guarantees better descriptions as we gather descriptive information about the outliers during the outlier detection process. Furthermore, we enhance the outlier ranking quality by considering only subspace projections that seem to be the reasons for highly deviating objects. Overall, we base our method on our adaptive outlier ranking as described in Chapter 13. For this outlier ranking, we propose a novel significance test for selecting only non-uniformly distributed subspaces in Sections 14.2.1, before we derive our descriptive components in Section 14.2.2.

14.2.1 Statistically selecting relevant subspaces

In this section we propose the selection of significantly non-uniformly distributed subspaces for our outlier ranking. As motivated in our toy example (cf. Sec. 14.1), the challenging task is to detect these *relevant subspaces* which can distinguish the outliers from regular objects. Hence, it is crucial to exclude some subspaces which hinder the distinction of outliers by uniformly random distributed objects. All objects seem outliers in these subspaces due to a missing contrast between outliers and regular objects. Any deviation measure which is based on distances would be unable to provide a meaningful degree of deviation as discussed in Challenge 2.

Thus, our general idea is to include only subspaces which are distributed significantly different then the uniformly random distribution. Hence, we exclude the subspaces that do not provide any distinction between objects and hinder our outlier detection. As first informal formulation we defined the set of relevant subspaces in Definition 43. According to this definition a subspace S is called relevant (for outlier ranking), iff S is not distributed uniformly random.

A key observation for relevant subspaces is that with increasing number of attributes in a subspace S, one reaches subspaces with uniformly distributed objects where outliers do not show up any more. By including more and more attributes distances between objects grow more and more alike [BGRS99]. Thus, the selection of relevant subspaces can be reduced to the selection of relevant attributes to be included in a given subspace projection S, as stated in the following corrollary:

Corrollary 1. Uniformly distributed subspaces
Let $S = \{d_1, \ldots, d_k\}$ be a subspace. Then it holds true:

$$S \text{ uniformly distributed} \Rightarrow \begin{array}{l} d_1 \text{ uniformly distributed} \\ \wedge \ldots \wedge \\ d_k \text{ uniformly distributed} \end{array}$$

Consequently, by testing each attribute d_i we can assure that no uniformly distributed subspace is included in the set of relevant subspaces RS. Moreover, we discard a subspace based on these insights as soon as at least one attribute is distributed uniformly random.

Selection by Significance Test We base on statistical tests to detect relevant subspaces by excluding uniformly distributed attributes from further consideration. We perform an incremental processing of the subspaces including in each step an additional attribute for the considered subspace S. By adding attribute d_i to S we check if objects are uniformly distributed in d_i. We call an attribute d_i *relevant* for outlier ranking, if objects are significantly non-uniformly distributed. In contrast to other attributes, a relevant attribute might be added to S while preserving the clustered regions of subspace S also in subspace $S \cup d_i$. Summing up, we detect meaningful outliers by searching subspaces consisting only of relevant attributes containing clustered regions from which outliers can deviate.

Furthermore, as we aim at detecting outliers that deviate from clustered objects in their local neighborhood we check uniformly distribution according to this neighborhood and not w.r.t. the entire subspace. We base our subspace neighborhood on the locality idea given by the k nearest neighbors [BKNS00]. Considering a subspace S a group of objects is defined as the subspace region around the object o as: $SR(o, S) = kNN_S(o)$ where $kNN_S(o)$ are the k nearest neighbors of o in S. Based on this definition, our outlier ranking is computed for each object w.r.t. its subspace region $SR(o, S)$, instead of the whole DB. Hence, the choice of relevant subspaces occurs strictly on the basis of the object locality. This is in contrast to subspace search approaches [PHL04], which provide global subspace estimations supporting clustering with interesting projections. Compared to such approaches, the main advantage of *DescOut* is the selection of locally relevant subspaces for each object taking local deviation into account.

Overall, based on statistical significance and locality of objects our novel selection of subspaces is defined in the following significance test:

Definition 49. Significance Test
For subspace region SR in subspace $S \cup d_i$ the hypotheses H_0 and H_1 for a given relevant subspace S are defined as:

$$\begin{aligned} H_0: &\ SR \text{ in subspace } S \cup d_i \text{ is uniformly random} \\ &\ (d_i \text{ is not relevant}) \\ H_1: &\ SR \text{ in subspace } S \cup d_i \text{ is non-uniformly random} \\ &\ (d_i \text{ is relevant}) \end{aligned}$$

Furthermore, the chosen statistical significance test ensures the first error to be lower than a given significance level α:

$$P(H_0 \text{ is rejected } | H_0 \text{ is true }) \leq \alpha$$

According to Definition 49 we iteratively include only those attributes which show significantly non-uniformly distribution in the neighborhood of o. As statistical tool for testing uniform distribution we use the Kolmogorov-Smirnov goodness of fit test for the uniform distribution [Ste70]. In recent mining tasks this test has shown good performance for subspace cluster detection [MS08]. Being able to accept significantly non-uniformly distributed attributes we can safely use these attributes as relevant subspaces for outlier mining. As significance level for the statistical hypothesis test we set $\alpha = 0.01$. Thus, the probability of wrongly rejecting the hypothesis H_0 (the subspace is uniformly distributed) is only 1%, i.e. for one out of hundred uniform subspaces the test will make an error and state that this subspace is relevant for outlier ranking. This is important for high quality outlier ranking as each uniformly distributed subspace would hinder the distinction of outliers using the derived ranking value.

Keeping the focus on the quality and not on the efficiency of our ranking, we only give a basic solution for our novel selection of subspaces. We choose a bottom-up processing of subspaces, as used in state-of-the-art subspace clustering approaches

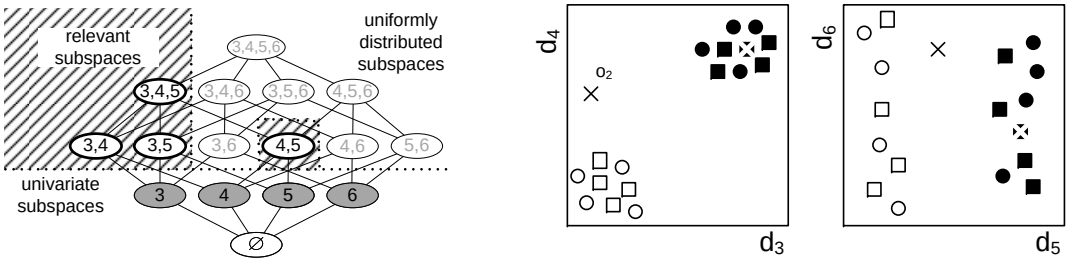
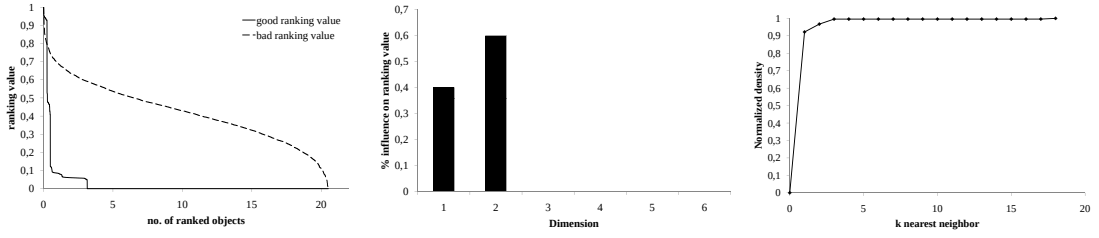


Figure 14.2: Relevant subspaces for outlier mining

[PHL04]. We start with few attributes and successively include more and more attributes. To illustrate the major effects of the included relevant subspace selection, we depict an example in Figure 14.2. *DescOut* starts by selecting a relevant subspace S (e.g. $\{d_3, d_4\}$) and collects the k nearest neighbors of o according to that projection, where the actual choice of k should be set conservatively high (≥ 20) in order to prevent imprecise subspace selections. Further relevant attributes are added successively (e.g. d_5). For our outlier ranking we only take the deviation factor of objects in these relevant subspaces into account. Please note, that the worst-case complexity of this basic processing as for all bottom-up approaches is exponential in the number of attributes. However in practical applications, by adding some attributes the subspaces become scattered (cf. subspace $\{d_3, d_4, d_6\}$). All objects seem to be outliers and thus we may stop processing these spaces and all further extensions of it. Using this pruning, we achieved a first solution for our model, which is applicable on benchmark data sets. However, further development of more enhanced algorithmic solutions is clearly required as future work. In this work, we focus on further enhancements of the ranking quality by providing additional descriptive components.

14.2.2 Descriptive components

Descriptive components are required for knowledge extraction in outlier mining. While automatic outlier detection methods provide a suggestion about possible outliers, final decision is typically made by domain experts. In addition to ranking values we provide descriptive components about relevant subspaces and deviation in local neighborhoods. As integrated approach, all of the required information for our descriptive components is gathered during the outlier ranking process. In the following we describe all of these components illustrated also for our running example in Figure 14.3.



(a) Ranking values (b) Influence of subspaces (c) Neighborhood density

Figure 14.3: Descriptive components for o_1 in our toy example

Ranking value based on deviation in subspaces In general, the ranking value has to provide a clear distinction between outliers and regular objects. As depicted in Figure 14.3(a), a "good ranking" achieves high ranking values for the few objects supposed to be outliers. Thus, they can be clearly distinguished by the rapid decrease to the regular objects showing low ranking values. Using full space or all possible subspaces for ranking would result in the depicted "bad ranking", where no clear distinction is possible. The slightly decrease in this ranking lacks a clear support for the outlier detection.

For our *DescOut* approach we use only the relevant subspace projections for each object. Thus, we achieve a clear drop in the ranking value. In our running example, only the two first subspace projections depicted in Figure 14.1 are selected as relevant subspaces for outlier ranking. Thus, both hidden outliers achieve high ranking values, while most of the residual objects do not show high deviation as they are clustered in one of the four dense regions.

Relevant subspaces We use our novel selection of relevant subspace projections not only for our ranking value, but also to derive one of our descriptive components. The relevant subspaces provide knowledge about the reasons why an object should be considered outlying. As depicted in Figure 14.3(b) for outlier o_1 we derive a histogram describing the relative contribution of each attribute to the overall ranking value. One can observe which are the most deviating attributes for the considered object. This additional knowledge can be used to verify each outlier or even to provide its outlying properties.

We derive the histogram iteratively while computing the ranking value. We initialize the histogram $H(o) = (H_1(o), \ldots H_d(o))$ for each attribute with $H_i(o) = 0$. For each relevant subspace S and outlierness measure $out(o, S)$ we then sum up the relative contributions in the respective histogram bins:

$$\forall i \in S : H_i(o) = H_i(o) + \frac{out(o, S)}{|S|}$$

The relative contribution $\frac{H_i(o)}{RV(o)}$ shown in the final histogram provides the attributes being most probably responsible for the measured outlier property.

Density Deviation in Local Neighborhood For outlier verification, one is not only interested in knowledge about the responsible attributes but also about the local neighborhoods of each outlier. Objects in these local neighborhoods appear as witnesses for the outlier properties as the outlier is highly deviating from this specific set of objects. By comparing the detected outlier with these objects one can extract the differences between a set of regular objects and one rare outlier.

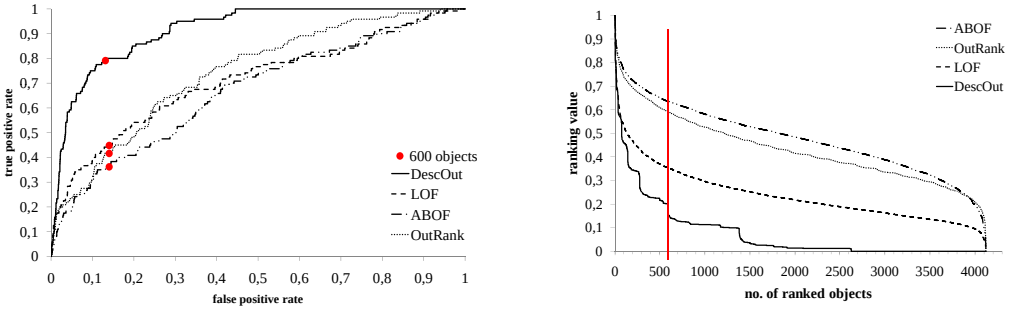
For comparison of outliers with their local neighborhood we plot the density of all objects in the subspace region $SR(o, S)$ as used in the significance test (cf. Sec. 14.2.1). A characteristic plot for a highly deviating object is depicted in Figure 14.3(c) where the outlier o_1 has very low density compared to the very high average density in its neighborhood.

14.3 Experiments

We evaluate the quality of *DescOut* on both synthetic and real world data. As competing approaches we choose LOF [BKNS00], its extension ABOF [KSZ08] and OUTRANK (cf. Chapter 12). As datasets we generate synthetic data where the hidden outliers are known in advance and as real world data we choose the pendigits data set from the UCI machine learning repository [AN07]. This data set allows us to verify detected outliers by visualization of the underlying digits. As quality measures we use true and false positive rates combined in the well-known ROC plot. In addition we show the contrast of our ranking value and present our descriptive components for real world data.

14.3.1 Synthetic Data

For experiments on synthetic data, we generate density-based clusters in arbitrary subspaces of the data. In addition, our generator adds numerous outliers by extracting some objects from their respective subspace clusters and deviating them in one or two relevant cluster dimensions. In our first experiment, this setup leads to the generation of 4121 objects in a 16d data space with clusters generated in 3d and 4d subspaces and 120 additional outliers in these subspaces.



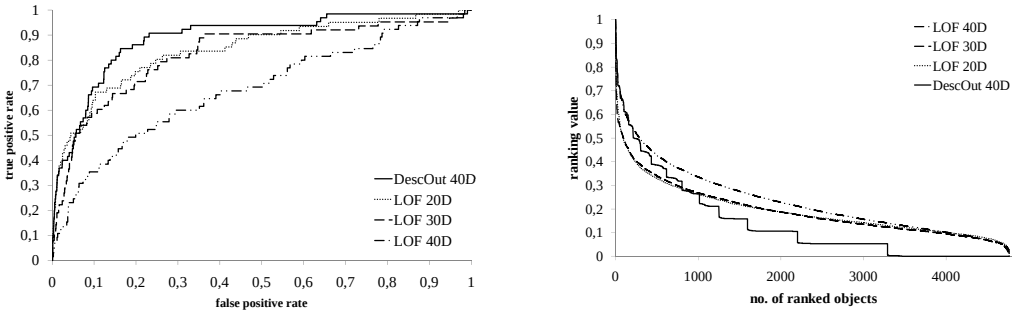
(a) ROC plot for outlier ranking

(b) Ranking values vs. objects

Figure 14.4: Outlier ranking on synthetic data

The resulting ROC and ranking value plot are depicted in Fig. 14.4. Obviously, the ROC plot shows that *DescOut* outperforms LOF, ABOF and OUTRANK. In the outlier ranking of *DescOut* the hidden outliers are detected earlier then in any other competing approach. Furthermore, we observe a rapid decrease in the ranking value of *DescOut* which is depicted in Fig. 14.4 by the red line at 600 objects. The rapid decrease of our ranking value shows us that deviation of following objects is by far lower giving us a hint about the residual outlier deviation. Thus, users might stopp further investigation at this point in the ranking if they are not interested in objects with such low deviation. At this point *DescOut* has detected almost 80% of the hidden outliers, while the competing approaches show less then 50% true positive rates in the ROC plot.

As traditional outlier detection is hindered by irrelevant attributes that lower the contrast between outliers and regular objects, we show the enhancement of our approach by the following experiment. We generated a 10d data set with outliers hidden in 3d and 4d subspaces. Additionally, we scale the dimensionality by adding 10, 20 and 30 irrelevant uniformly distributed attributes to the original data set. In Fig. 14.5 we show the results for three runs of LOF as best representative of the competing approaches, while *DescOut* is only given for one data set. We skip all other runs of *DescOut*, as they show even better ROC and ranking values. Overall, *DescOut* outperforms the competing approaches. Considering LOF with increasing number of attributes, quality drops as outliers are not top-ranked any more. In contrast, *DescOut* scales well providing a high quality outlier ranking.



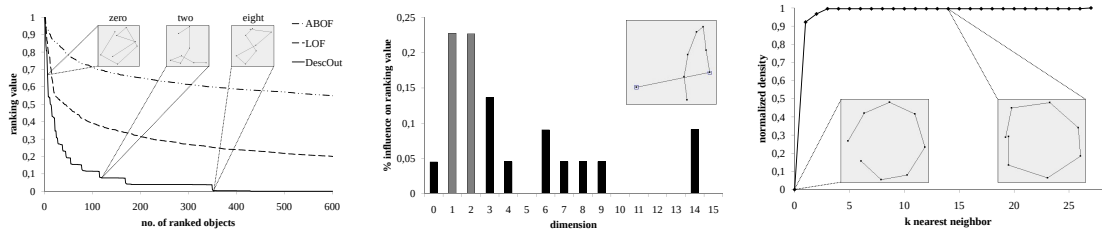
(a) ROC plot

(b) Ranking values

Figure 14.5: Enhancement w.r.t. dimensionality of data

14.3.2 Real World Data

We evaluate *DescOut* on the pendigits data set providing 7494 handwriten digits in a 16d space [AN07]. Since no outliers are labeled for this data set, we verify the outlierness of delivered objects and the correctness of our descriptive components by visualization. The respective plots are depicted in Fig. 14.6(a). Unfortunately, OUTRANK did not work on this data set due to the high runtimes for the underlying subspace clustering. For LOF and ABOF we show low quality in outlier ranking, similar to the observations on synthetic data. For *DescOut* the top-ranked objects are correct detected outlier. They appear to be not even numbers as the representatively chosen zero shows. For objects with significantly lower ranking values, *DescOut* returns correctly written digits. Taking a closer look at the reasons of outliers we inspect the relevant subspaces. The histogram for an outlying "one" in Fig. 14.6(b) shows the first two dimensions as highly deviating. In the visual representation this is confirmed by the wrong beginning of the digit highlighted by black squares. In Fig. 14.6(c) the nearest neighbors of an outlying zero are shown with their corresponding normalized densities. Obviously, all surrounding objects have much higher densities and can be visualized as very ordinary zeroes, as the representatively chosen zero illustrates.



(a) Ranking Values vs. first 600 ranked objects

(b) Outlying one

(c) Outlying zero in subspace $\{5, 6\}$

Figure 14.6: Descriptive outlier ranking on Pendigits data set

14.4 Enhancements by descriptive outlier ranking

We introduced the *DescOut* approach providing a descriptive outlier ranking. It overcomes two major drawbacks of existing outlier ranking approaches: lack of descriptive components providing reasons of outlier properties and low quality of outlier ranking for outliers hidden in subspace projections. In our *DescOut* approach, we detect for each object subspace projections describing the reasons for its outlier properties. We enhance outlier ranking by considering object deviation only in relevant subspaces. In addition, we provide these relevant attributes and local neighborhoods of each object as witnesses for outlier properties. Thorough experiments demonstrate that *DescOut* clearly outperforms existing outlier ranking algorithms in detection of outliers, while providing meaningful descriptive information about their outlier properties.

The extracted knowledge in our descriptive components can be used for interactive exploration of the detected outliers. Users may interact with the resulting descriptive outlier ranking by providing specific attributes which are supposed to be the reasons for outliers or focusing on specific sets of suspicious objects. This leads to in-depth analysis of the detected outliers. Overall, our descriptive outlier ranking covers the major steps of the KDD process. It provides data analysis techniques for the detection of outliers while our novel descriptive components support the user with additional visual information for the final knowledge extraction.

Part V

Summary

Chapter 15

Conclusion and future work

Overall, the research work presented in this thesis has an impact in many different areas. In the following, we summarize the research results for knowledge discovery in subspace projections, but also its applications to several domains investigated in different projects. Finally, we highlight some of the most promising future research directions derived out this thesis.

15.1 Research results for knowledge discovery in subspaces

Most important, this thesis provides concise research results in the area of knowledge discovery in databases. As various recent applications provide high dimensional databases where patterns are hidden in subsets of the given attributes, we focus on clustering and outlier detection in subspace projections. Our research ranges from novel data analysis models up to evaluation and exploration techniques for the mining results.

In Part I we focus on enhanced clustering models ensuring high quality subspace clustering results. We propose a novel unbiased subspace cluster definition for the density-based clustering paradigm. Adapting to the dimensionality of the considered subspace our unbiased density measures ensure comparable density for arbitrary subspaces. Our first model has been published at the 2007 IEEE international conference on data mining [AKMS07a]. It proposes a variable density threshold adapting to the expected density in each subspace. Furthermore, as major challenge for subspace clustering we tackle the redundancy of subspace cluster results. Having to cope with many redundant subspace clusters traditional methods fail in detecting high quality results. Such a high quality result should contain a set of non-redundant patterns, where each cluster contributes to the extracted knowledge. In this thesis we propose several techniques tackling this general challenge of subspace clustering. Our DUSC model is the first subspace clustering approach coping with redundancy in subspace clustering. While it proposes a simple redundancy definition, taking only a pairwise comparison of clusters into account, our more enhanced optimization techniques propose a global optimization of most interesting and non-redundant subspace clusters.

Accepted for publication at the 2009 IEEE international conference on data mining [MAG+09b], our relevant subspace cluster definition ensures to detect all and only the most relevant subspace clusters. For this global optimization based on the overall set of subspace clusters we have proven that computation of such a complex model is NP-hard. In our most recent subspace clustering model we propose an orthogonal subspace clustering that actively searches for multiple hidden concepts for each data object. While clusters in similar subspaces have to be considered as redundant information, orthogonal subspaces provide addition knowledge about the data. Presented at 2009 ACM international conference on information and knowledge management [GMFS09], this model ensures high quality results, but, as our previous optimization model, it is proven to be NP-hard. Applying our orthogonal subspace clustering model to a given clustering solution one can even detect alternative subspace clusters as described in our most recent publication [GFMS10]. Overall, the proposed optimization models are general approaches for non-redundant or orthogonal subspace clustering. Although we focus our research on the density-based paradigm as underlying cluster model our research provides a general contribution for many subspace cluster definitions.

Focusing not only on high quality results but also on efficient computation, Part II proposes several techniques for efficiency improvement. Tackling both the high cost of database access and the exponential search space, our techniques ensure scalability to large and high dimensional databases. We propose a multi-step architecture with several efficient filters to reduce database access. Presented at 2008 ACM international conference on information and knowledge management [AKMS08a], our technique achieves both efficient computation and lossless detection of all subspace clusters according to our unbiased subspace cluster definition. Further efficiency improvements were achieved by our in-process removal of redundancy presented at the 2008 IEEE international conference on data mining [AKMS08b]. Using this technique one can exclude large parts of the exponential search space as they contain only redundant clusters. Furthermore, by extending our novel index support we propose to unify density-based subspace clustering and frequent itemset mining for an efficient mining of heterogeneous data. Presented at the 2009 international conference on statistical and scientific database management [MAS09], this unification is important for application scenarios where arbitrary attribute types have to be analyzed. As part of this unification, we provide a thorough comparison between frequent itemset and subspace clustering as different mining paradigms and derive common properties for their processing. For our optimization models, such an efficient and complete cluster detection as presented by our first processing schemes seems not possible as both are NP-hard problems. We focus on an approximate solution for both models. According to these models not all subspace clusters are relevant. Thus, for a high quality and efficient computation one might skip large parts of the search space and process only most promising dense subspace regions. In our publication at 2009 SIAM international conference on data mining [MAG+09a], we propose a general density estimation technique showing both high quality estimates and efficient computation with only few database scans. Based on this method we propose a general steering

of subspace clustering for further efficiency improvement, currently submitted for publication. As described in Chapter 9 we propose efficient information gathering and feedback from progressively detected subspace clusters to steer the processing to only few but most promising subspace regions.

In Part III, we propose a more general contribution to the research community. While previous parts focus on specific enhancements of subspace clustering techniques, our systematic evaluation study provides a thorough comparison of major paradigms proposed in the last decade. We highlight the differences of subspace clustering and projected clustering models and show their effects in a thorough experimental evaluation. We use a broad set of synthetic and real world benchmark databases for our evaluation and compare each approach with various quality measures. Overall, our evaluation presented at 2009 international conference on very large data bases [MGAS09] is a major contribution to the young research field of subspace clustering. Furthermore, we contribute to the repeatability initiative by providing an open source framework for subspace clustering. Initiated by SIGKDD conference, repeatability of experimental results is becoming a major issue in the KDD community. Authors are encouraged to provide implementations and data sets. However, for overall repeatability and comparability of experimental results it is essential to have open source implementations of a broad set of algorithms at hand. Our OpenSubspace framework enables comparability based on a common implementation framework. Providing a broad set of approaches and evaluation measures OpenSubspace provides the basis for thorough evaluations in future research. Several parts of this framework have been presented as demonstration systems at conferences [MAK+08, AMK+08] and as framework proposal at the 2009 open source in data mining workshop [MAG+09a]. This evaluation part is of major importance for our evaluation study, while for interactive exploration of subspace clustering we have published our visualization techniques in ACM SIGKDD explorations special issue on visual analytics [AKMS07b]. Overall, a short summery of our work in the area of subspace clustering is provided in two brief surveys focusing on results out of this thesis [MÏ0, AMG+10].

In addition to our research on subspace clustering, Part IV proposes several techniques for outlier detection in subspace projections. As essential task in application scenarios like fraud detection or anomaly detection, outlier mining has to cope with objects deviating only in some subspace projections. As basic approach we propose to use high quality clustering results provided by our subspace cluster models for outlier detection. We developed novel scoring functions that rank objects according to their degree of deviation in subspace projections. Our first work on outlier mining has been published at 2008 international workshop on ranking in databases [MASS08], where we focus on outlier mining as a post-processing to subspace clustering. Further extensions of this work, considering direct outlier detection have been accepted for publication at 2010 ACM international conference on information and knowledge management [MSS10]. As described in Chapter 13 and 14 we propose an adaptive outlier deviation measuring outlierness of objects in some relevant subspace projections. For selection of these subspace projections we propose a statistical significance

test such that only significantly non-uniformly distributed spaces are used for deviation measurement. In addition to our novel outlier ranking we develop descriptive components as witnesses for the outlier properties. Both outlier ranking and descriptive components can be explored by the user to extract the desired knowledge about possible outliers in the database as described in our exploration toolkit for subspace outlier ranking [MSG+10]. Overall, as for subspace clustering we provide concise research results not only for detection of patterns but also for their evaluation and exploration covering the major parts of the KDD process.

15.2 Application scenarios for proposed techniques

As research results in this thesis are applicable on arbitrary high dimensional databases, we have applied them in several collaborative projects. We either used the proposed techniques or specialized variants of these approaches to extract knowledge out of real world databases provided by our collaboration partners in these projects.

Bioinformatics The domain of Bioinformatics has shown to be the most common application scenario for subspace clustering. Biological experiments produce large amounts of data, e.g. in high throughput experiments with gene expression arrays. Such experiments are used to identify similar behavior of genes on a subset of experimental conditions. While gene expression analysis has been an early application of subspace clustering, we focused in one of our first projects on *comparative genomic hybridization* databases provided by a collaboration with pathologists. As proof of concept we showed that applying our subspace clustering techniques on such data results in meaningful clusters for the domain experts. In a second project in collaboration with *Cell Biology*, clustering and outlier mining methods where used for enhancing the labeling of biological experiments. While our collaboration partners provided an automatic text mining tool for labeling biological experiments our mining of gene expression levels was able to assist in detecting mislabeled objects or propagating labels to unlabeled objects [RKM+08]. Overall, both projects have shown the applicability of our research in the area of Bioinformatics, but also raised new questions on handling of missing values in subspace clusters. Derived out of an application, handling of missing values is now part of our ongoing research on subspace cluster definitions.

Data mining on sensor measurements In some of our external collaborations with companies (not to be named explicitly) we applied data mining in subspaces of sensor measurements. One of these companies is aiming at outlier detection on high dimensional heterogeneous senor measurements. In collaboration with domain experts we developed a novel mining approach for this task. Abstracting from the used data in this application scenario we enhanced our subspace clustering models as described in Chapter 7. This project shows that our approaches do not only provide high quality results on benchmark databases but have also been successfully applied to real world problems. Further requirements where proposed by a second

company using our subspace clustering on sensor measurements for fault detection in production chains. While our subspace clustering results produce only groupings of similar objects our ongoing work focuses on extraction of additional correlation rules. These rules should provide reasons for different faults and assist the domain experts in finding reasons for faults observed in the production chain. Both of these collaborations have lead to novel knowledge extracted by our approaches for the companies. Furthermore, they both raised interesting application oriented research questions.

Energy efficiency in mobile networks In a collaboration between computer science and electrical engineering we developed specialized mining approaches as part of the research cluster UMIC on *Ultra Highspeed Mobile Information and Communication*. Aiming at next generation mobile communication the research cluster has its focus also on data provisioning. Gathering measured information from many mobile clients is especially challenging due to the limited energy resources on mobile devices. With our research on data analysis of these measurements we focus on energy efficient communication. Detecting groups of clients measuring similar information leads to reduced communication by aggregated data transmission. Increasing the lifetime of the overall network is of major importance for data provisioning. Our specialized clustering techniques have shown to reduce energy consumption significantly [HMS09, HMS+10]. One of the most promising future research directions in this project is the extension of subspace clustering to stream databases detecting subspace clusters that evolve over time. Motivated by this project, further interesting research questions arise if one includes network data into the data mining process. The underlying communication or social network is of major importance for such applications and should be considered in future research.

Incremental data analysis In the most recent collaboration, high dimensional data is provided by marine scientists from *Alfred Wegener Institute*. The general aim is to identify regions in the ocean that show similar measurements in a subset of attributes. Having detected these subspace clusters they have to be traced over time such that domain experts can extract knowledge out of these evolving regions. We base on this real world database for future experiments but also try to abstract from this scenario to derive general mining solutions. The application scenario can be abstracted to location aware sensor nodes measuring high dimensional data. In general, such data is typically provided by environmental surveillance projects not limited to ocean databases (e.g. also for alpine surveillance in the Swiss Experiment[1]). General challenges arise out of such application scenarios for incremental data analysis. While our research has focused on static databases, sensor measurements require evolving models. Incremental data analysis should base the next mining step on the previous results. Aiming at efficient and high quality approaches, an incremental processing of subspace clusters may cope with both the high data rates of sensor measurements and the tracing of evolving clusters over time.

[1] http://www.swiss-experiment.ch/

15.3 Future work

Further research questions arise out of our development of novel data mining approaches but also due to the mentioned work on applications in various projects. Let us summarize the most promising challenges for future research in this area.

Extending orthogonal subspace clustering paradigm In general, our research work in the area of orthogonal concept detection has the highest potential for future subspace clustering models. Combining both the high quality of non-redundant subspace clustering with the detection of multiple concepts our model tackles all proposed challenges. As multiple concepts arise in various recent applications where objects can be clustered due to several reasons this model seems the most promising for future research in this area. As an emerging research direction we have a tutorial in this area on "Discovering Multiple Clustering Solutions" [MGFS10]. It highlights several novel challenges in this research field. Especially, the following two challenges should be tackled in general clustering models and domain specific instantiations.

First, one has to tackle novel challenges for the general orthogonal clustering model. As our models contain cluster oriented definitions, they provide several parameters to set the properties of desired clusters (e.g. minimum density threshold). In contrast, clustering oriented approaches provide a fixed number of cluster k as a parameter. This seems a very hard restriction to many k-means based approaches. However, clustering oriented approaches have major benefits compared to an uncontrolled handling of the result size which may lead to huge amounts detected clusters. With our non-redundant clustering definitions we have done major steps for unifying both notions. However, we do not provide parameters to set the number of clusters or the number of multiple concepts to be detected per object. In some domains such parameters might be appropriate. A fundamental question is how to introduce such parameters to enhance our models. Such parameters may either be derived out of our models or included in more abstract variants of our subspace clustering definitions. Furthermore, providing such parameters seems to solve some questions about termination of multi concept detection. However, more important it poses novel challenges for the subspace clustering model. Currently, the proposed optimization techniques are only aware of the already detected clusters. They do not know anything about the probable clusters that might be detected in the future processing. Extending clustering models based on a future aware processing is a key requirement for future research.

Second, challenges arise for the enhancement of our general techniques in specific domains. Most promising domains are multi dimensional streams and multi-labeled graph databases. Streams are an important field of research in sensor networks where we do not have one static database. Incrementally clustering such high dimensional streams poses novel challenges for subspace clustering models. In contrast to static results one has to cope with subspace clusters that evolve over time. Multi-labeled graph structures arise out of recent applications such as communication networks or social networks. Analysis of such networks is of great importance for interdisci-

plinary sciences and sooner than expected for commercial applications. For tackling multiple labels and node connections as novel heterogeneity in the data, subspace clusters might consider density-connected subgraphs as subspaces in the graph related attributes. Overall, in each of these domains additional specific challenges arise. However, the general challenges discussed in this thesis such as adaptation to subspace projections and redundancy in the result set have to be tackled as well. In combination the specific and general challenges are very interesting questions for future research. How should models be defined in these domains and how should they cope with the additional efficiency challenge of "time" and "graph structure" which add an additional complexity dimension to the problem?

Future aware steering by estimated results as potential efficiency improvement For all approaches mentioned in this thesis we had to cope with both quality and efficiency challenges. The key of efficiency in subspace clustering is possible pruning which can be performed as soon as possible. While traditional approaches where more or less blind they used only local information such as "is this subspace region dense or not" for pruning. This results in highly inefficient processing. All of our methods enhanced pruning by including additional information as for in-process pruning of redundancy: "Does this subspace region contain a non-redundant cluster w.r.t. the already detected clustering result?" Further efficiency enhancements may be possible by including information about possible clusters in the future processing. Estimating such future results might be an additional task that can be tackled by our general density estimation technique to solve this problem. Future pruning could base on question like this: "As I have estimated too many resulting clusters with these parameters, should I exclude more concepts by looking into more orthogonal subspaces?" Such a self-adaptive algorithm would yield an easier way of parametrization, higher quality of results and most probably a significant efficiency improvement.

Enhanced evaluation measures for subspace clustering In our evaluation study we have compared a broad set of approaches from different clustering paradigms. For a fair comparison we used various quality measures proposed in the literature. We also started development of our own evaluation measures coping with specific quality properties like non-redundant results. To ensure an objective evaluation we have compared the main characteristics of all measures in our evaluation study and published the results of each measure in all experiments. However, none of the proposed measures seems suitable as a unified quality measure to be used as a benchmark in future evaluations. Currently, for each measure one may create counter examples where the clustering is clearly non-optimal while the measures show high quality ratings. Similar to our evaluation study on clustering approaches, an evaluation study on quality measures is of great importance for the research community. Defining theoretical quality properties and deriving out of these properties a benchmark measure could be part of such a study. Using such a single unified measure would result in fair comparison in future scientific publications. As first steps in this direction, we have summarized open challenges for evaluation of multiple clusterings [FGK+10].

Iterative processing schemes for subspace clustering In this work we show high quality approaches for static databases with runtimes of some minutes on benchmark data. However, for an interactive exploration of large and high dimensional real world databases one requires incremental approaches producing some rough approximate results in seconds to provide users an overview of possible subspace clusters. Users may provide feedback based on these results to steer the clustering algorithm w.r.t. their own interests. In general, such interactive processing requires subspace clustering techniques that provide results on multiple scales from rough approximations to high quality detailed clusterings. Results are iteratively enhanced by clustering without restarting the processing. User feedback can be used in addition to automatic steering heuristics such that most interesting clusters are detected first. The orthogonal concepts to these clusters or different subspace regions are processed afterwards. Overall, this research would open a new paradigm of semi-automatic subspace clustering approaches.

Symbiosis of outlier and cluster detection in subspace projections In this work we considered cluster and outlier detection separately showing significant improvements in both areas. In both areas we are aware of each other such that we can gain some benefits out of outlier awareness in clustering and cluster detection in outlier mining. On the one side, we base our subspace clustering methods on the well established density-based paradigm. Furthermore, in contrast to projected clustering approaches our novel optimization methods do not force the subspace clustering to cover all objects. Thus, all proposed approaches are aware of outliers in the database, resulting in high clustering quality even for noisy data. On the other side we propose a visionary post-processing schema extracting outliers out of subspace clustering results. Even our more enhanced outlier ranking methods are motivated by challenges that we observed in prior work for subspace clustering. Further symbiotic improvements in both areas are clearly of major interest. Especially, the recent orthogonal subspace clustering models opens new challenges for outlier detection. Outlier detection that is aware of these multiple concepts might detect objects that are clustered in almost all provided concepts, but clearly deviates in one concept. As most promising area we observe the development of novel scoring functions that are aware of our enhanced subspace clustering models. Even the reason why an object is highly deviating is provided by the one missing concept. For clustering on the other side our statistical significance tests might be of interest for efficient exclusion of uniformly random subspaces. One could develop further approximate filter steps based on this idea derived from our outlier mining research.

Summing up, in this work we proposed enhanced models, efficient solutions, evaluation and exploration techniques for clustering and outlier mining in subspace projections. All of this forms a major contribution to this young research field. Furthermore, we derived a significant amount of knowledge about subspace mining in general which opens some interesting research questions for further development.

Bibliography

[AFP09] F. Angiulli, F. Fassetti, and L. Palopoli. Detecting outlying properties of exceptional objects. *ACM Transactions on Database Systems (TODS)*, 34(1):1–62, 2009.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[AKMS07a] I. Assent, R. Krieger, E. Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 409–414, 2007.

[AKMS07b] I. Assent, R. Krieger, E. Müller, and T. Seidl. VISA: Visual subspace clustering analysis. *ACM SIGKDD Explorations*, 9(2):5–12, 2007.

[AKMS08a] I. Assent, R. Krieger, E. Müller, and T. Seidl. EDSC: Efficient density-based subspace clustering. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1093–1102, 2008.

[AKMS08b] I. Assent, R. Krieger, E. Müller, and T. Seidl. INSCY: Indexing subspace clusters with in-process-removal of redundancy. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 719–724, 2008.

[AKZ08] E. Achtert, H.-P. Kriegel, and A. Zimek. ELKI: A software system for evaluation of subspace clustering algorithms. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 580–585, 2008.

[AMG+10] I. Assent, E. Müller, S. Günnemann, R. Krieger, and T. Seidl. Less is more: Non-redundant subspace clustering. In *Proceedings of the International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[AMK+08] I. Assent, E. Müller, R. Krieger, T. Jansen, and T. Seidl. Pleiades: Subspace clustering and evaluation. In *Proceedings of the European*

Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), pages 666–671, 2008.

[AN07] A. Asuncion and D. J. Newman. UCI machine learning repository http://www.ics.uci.edu/∼mlearn/MLRepository.html. University of California, Irvine, School of Information and Computer Sciences, 2007.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 487–499, 1994.

[AWY+99] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park. Fast algorithms for projected clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 61–72, 1999.

[AY00] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 70–81, 2000.

[AY01] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 37–46, 2001.

[BCD+09] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Koetter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. Knime: The Konstanz Information Miner. *ACM SIGKDD Explorations*, 11(1):26–31, 2009.

[BFH95] Y. M. Bishop, S. E. Fienberg, and P. W. Holland. *Discrete multivariate analysis: Theory and Practice*. MIT Press, Cambridge, 1995.

[BGRS99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful. In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 217–235, 1999.

[BKKK04] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 27–34, 2004.

[BKNS00] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

[BL94] V. Barnett and T. Lewis. *Outliers in statistical data*. John Wiley, 1994.

[BS04] S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 19–26, 2004.

[BZ07] B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 63–72, 2007.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[CFD07] Y. Cui, X. Z. Fern, and J. G. Dy. Non-redundant multi-view clustering via orthogonalization. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 133–142, 2007.

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

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[Dv03] S. Dvzeroski. Multi-relational data mining: an introduction. *ACM SIGKDD Explorations*, 5(1):1–16, 2003.

[DZL04] J. Demsar, B. Zupan, and G. Leban. Orange: From experimental machine learning to interactive data mining, white paper (www.ailab.si/orange), faculty of computer and information science, university of ljubljana, (2004).

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[FGK+10] I. Färber, S. Günnemann, H.-P. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. On using class-labels in evaluation of clusterings. In *Proceedings of the International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[FMW08] P. Filzmoser, R. Maronna, and M. Werner. Outlier identification in high dimensions. *Computational Statistics and Data Analysis*, 52(3):1694–1711, 2008.

[Fre08] Frequent Itemset Mining Dataset Repository. http://fimi.cs.helsinki.fi/, 2008.

[GFMS10] S. Günnemann, I. Färber, E. Müller, and T. Seidl. ASCLU: Alternative subspace clustering. In *Proceedings of the International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust) in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[GH06] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), 2006.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP- Completeness*. W. H. Freeman, San Francisco, 1979.

[GMFS09] S. Günnemann, E. Müller, I. Färber, and T. Seidl. Detection of orthogonal concepts in subspaces of high dimensional data. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1317–1326, 2009.

[GOP04] A. Ghoting, M. E. Otey, and S. Parthasarathy. LOADED: Link-based outlier and anomaly detection in evolving data sets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 387–390, 2004.

[HK98] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 58–65, 1998.

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

[HLP88] G. H. Hardy, J. E. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, 1988.

[HMS09] M. Hassani, E. Müller, and T. Seidl. EDISKCO: Energy efficient distributed in-sensor-network k-center clustering with outliers. In *Proceedings of the International Workshop on Knowledge Discovery from Sensor Data (SensorKDD) in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48, 2009.

[HMS+10] M. Hassani, E. Müller, P. Spaus, A. Faqolli, T. Palpanas, and T. Seidl. Self-organizing energy aware clustering of nodes in sensor networks using relevant attributes. In *Proceedings of the International Workshop on Knowledge Discovery from Sensor Data (SensorKDD) in conjunction*

with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 87–96, 2010.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2000.

[HXD03] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[Jol86] I. Joliffe. *Principal Component Analysis*. Springer, New York, 1986.

[JTH01] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 293–298, 2001.

[Kei02] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[KKK04] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 246–257, 2004.

[KKKW03] K. Kailing, H.-P. Kriegel, P. Kröger, and S. Wanka. Ranking interesting subspaces for clustering high dimensional data. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 241–252, 2003.

[KKRW05] H.-P. Kriegel, P. Kröger, M. Renz, and S. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 250–257, 2005.

[KKSZ09] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Proceedings of the Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, pages 831–838, 2009.

[KKZ09] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1), 2009.

[KN99] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 211–222, 1999.

[KNT00] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *VLDB Journal*, 8(3):237–253, 2000.

[KSZ08] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452, 2008.

[KWX+06] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 882–893, 2006.

[KXWR06] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage, 2006.

[LK05] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 157–166, 2005.

[MÜ10] E. Müller. Mining subspace clusters: Enhanced models, efficient algorithms and an objective evaluation study. In *Proceedings of the PhD Workshop at the International Conference on Very Large Data Bases (VLDB)*, 2010.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[MAG+09a] E. Müller, I. Assent, S. Günnemann, T. Jansen, and T. Seidl. OpenSubspace: An open source framework for evaluation and exploration of subspace clustering algorithms in WEKA. In *Proceedings of the Open Source in Data Mining workshop in conjunction with Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 4–15, 2009.

[MAG+09b] 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 *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 377–386, 2009.

[MAK+08] E. Müller, I. Assent, R. Krieger, T. Jansen, and T. Seidl. Morpheus: Interactive exploration of subspace clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1089–1092, 2008.

[MAK+09] E. Müller, I. Assent, R. Krieger, S. Günnemann, and T. Seidl. DensEst: Density estimation for data mining in high dimensional spaces. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 173–184, 2009.

[MAS09] E. Müller, I. Assent, and T. Seidl. HSM: Heterogeneous subspace mining in high dimensional data. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 497–516, 2009.

[MASS08] E. Müller, I. Assent, U. Steinhausen, and T. Seidl. Outrank: Ranking outliers in high dimensional data. In *Proceedings of the International Workshop on Ranking in Databases (DBRANK) in conjunction with IEEE International Conference on Data Engineering*, pages 600–603, 2008.

[MGAS09] E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2(1):1270–1281, 2009.

[MGFS10] E. Müller, S. Günnemann, I. Färber, and T. Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *Tutorials at IEEE International Conference on Data Mining (ICDM)*, 2010.

[MMK+05] V. Markl, N. Megiddo, M. Kutsch, T. M. Tran, P. J. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 373–384, 2005.

[MS08] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 533–541, 2008.

[MSE06] G. Moise, J. Sander, and M. Ester. P3C: A robust projected clustering algorithm. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 414–425, 2006.

[MSG+10] E. Müller, M. Schiffer, P. Gerwert, M. Hannen, T. Jansen, and T. Seidl. SOREX: Subspace outlier ranking exploration toolkit. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 607–610, 2010.

[MSS10] E. Müller, M. Schiffer, and T. Seidl. Adaptive outlierness for subspace outlier ranking. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2010.

[NGC01] H. Nagesh, S. Goil, and A. Choudhary. Adaptive grids for clustering massive data sets. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2001.

[Pea00] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[PHIS96] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. *ACM SIGMOD Record*, 25(2):294–305, 1996.

[PHL04] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations*, 6(1):90–105, 2004.

[PHM00] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[PJAM02] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 418–427, 2002.

[PK01] T. Palpanas and N. Koudas. Entropy based approximate querying and exploration of datacubes. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 81–90, 2001.

[PM06] A. Patrikainen and M. Meila. Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(7):902–916, 2006.

[QD09] Z. Qi and I. Davidson. A principled and flexible framework for finding alternative clusterings. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 717–726, 2009.

[RKM+08] D. Ruau, C. Kolarik, H.-T. Mevissen, E. Müller, I. Assent, R. Krieger, T. Seidl, M. Hofmann-Apitius, and M. Zenke. Public microarray repository semantic annotation with ontologies employing text mining and expression profile correlation. *BMC Bioinformatics*, 9(S-10), 2008.

[RN03] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[Spe87] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101, 1987.

[ST96] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, 1996.

[Ste70] M. Stephens. Use of the Kolmogorov-Smirnov, Cramer-von Mises and related statistics without extensive tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 115–122, 1970.

[SZ04] K. Sequeira and M. Zaki. SCHISM: A new approach for interesting subspace mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 186–193, 2004.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, USA, 2005.

[WHB10] B. Wiswedel, F. Höppner, and M. Berthold. Learning in parallel universes. *Data Mining and Knowledge Discovery*, 21(1):130–152, 2010.

[Whi90] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, 1990.

[Wil08] G. Williams. Rattle: A graphical user interface for data mining in R using GTK. R package version 2.3.115. http://rattle.togaware.com/, 2008.

[YM03] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 689–692, 2003.

[Zak00] M. Zaki. Generating non-redundant association rules. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2000.

[ZYH+07] F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng. Mining Colossal Frequent Patterns by Core Pattern Fusion. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 706–715, 2007.

Lebenslauf – Curriculum Vitae

Persönliche Daten

Dipl.-Inform. Emmanuel Alexander Müller
Geboren am 13.04.1982 in Athen, Griechenland
Nationalität: deutsch, griechisch

Schloss-Schönau-Str. 68A
52072 Aachen
Tel.: 0241 53101590
Mail: Emmanuel.Mueller@rwth-aachen.de

Ausbildung

Seit 2007	Promotionsstudium Informatik an der RWTH Aachen
2007	Abschluss mit Auszeichnung als Diplom Informatiker
2004	Abschluss des Vordiploms mit der Note Sehr Gut
2002 – 2007	Studium Informatik (Nebenfach Biologie) an der RWTH Aachen
2001 – 2002	Fernstudieneinstieg Informationstechnik an der TU Kaiserslautern
2001	Abschluss Allgemeine Hochschulreife
1992 – 2001	Megina-Gymnasium in Mayen
1988 – 1992	Grundschule Hinterburg in Mayen

Berufserfahrung / Grundwehrdienst

Seit 2007	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Informatik 9 der RWTH Aachen
2005	Praktikum am Fraunhofer-Institut FIT in St. Augustin
2003 – 2006	Studentische Hilfskraft an Mathematik- und Informatik-Lehrstühlen der RWTH Aachen
2001 – 2002	Grundwehrdienst im Fernmelderegiment 920 in Kastellaun
1999 – 2000	Freier Mitarbeiter als Programmierer für audiofind.de
1998 – 2004	Freier Mitarbeiter als Vermesser für E.K.S.–Tiefbau GmbH

Sprachkenntnisse

Deutsch und Griechisch als Muttersprache
Englisch und Französisch als Fremdsprache